Adapting Large Language Models for Education: Foundational Capabilities, Potentials, and Challenges

QINGYAO LI and LINGYUE FU, Shanghai Jiao Tong University, China
WEIMING ZHANG and XIANYU CHEN, Shanghai Jiao Tong University, China
JINGWEI YU, Shanghai Jiao Tong University, China
WEI XIA†, Huawei Noah's Ark Lab, China
WEINAN ZHANG†, Shanghai Jiao Tong University, China
RUIMING TANG, Huawei Noah's Ark Lab, China
YONG YU†, Shanghai Jiao Tong University, China

Online education platforms, leveraging the internet to distribute education resources, seek to provide convenient education but often fall short in real-time communication with students. They often struggle to address the diverse obstacles students encounter throughout their learning journey. Solving the problems encountered by students poses a significant challenge for traditional deep learning models, as it requires not only a broad spectrum of subject knowledge but also the ability to understand what constitutes a student's individual difficulties. It's challenging for traditional machine learning models, as they lack the capacity to comprehend students' personalized needs. Recently, the emergence of large language models (LLMs) offers the possibility for resolving this issue by comprehending individual requests. Although LLMs have been successful in various fields, creating an LLM-based education system is still challenging for the wide range of educational skills required. This paper reviews the recently emerged LLM researches related to educational capabilities, including mathematics, writing, programming, reasoning, and knowledge-based question answering, with the aim to explore their potential in constructing the next-generation intelligent education system. Specifically, for each capability, we focus on investigating two aspects. Firstly, we examine the current state of LLMs regarding this capability: how advanced they have become, whether they surpass human abilities, and what deficiencies might exist. Secondly, we evaluate whether the development methods for LLMs in this area are generalizable—that is, whether these methods can be applied to construct a comprehensive educational supermodel with strengths across various capabilities, rather than being effective in only a singular aspect. Based on the current development status, we further outline two approaches for an LLM-based education system: a unified approach and a mixture-of-expert (MoE) approach. Finally, we explore the challenges and future directions, providing new research opportunities and perspectives on adapting LLMs for education.

CCS Concepts: • **Information systems**; • **Applied computing → Education**;

Additional Key Words and Phrases: Large Language Models, Educational Data Mining

ACM Reference Format:
Qingyao Li, Lingyue Fu, Weiming Zhang, Xianyu Chen, Jingwei Yu, Wei Xia†, Weinan Zhang†, Ruiming Tang, and Yong Yu†. 2018. Adapting Large Language Models for Education: Foundational Capabilities, Potentials, and Challenges. In *Proceedings of Make sure to*

† Weinan Zhang, Yong Yu and Wei Xia are the co-corresponding authors.

Manuscript submitted to ACM

enter the correct conference title from your rights confirmation emai (Conference acronym 'XX). ACM, New York, NY, USA, 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Education plays a vital role in shaping individuals' futures as it forms the foundation for providing people with knowledge, skills, and critical thinking abilities. Conventional education systems heavily rely on teachers for imparting knowledge to students, which place a significant demand on educational resources. However, the advent of online education has substantially lowered the cost of accessing these educational materials. Many people are conveniently acquiring knowledge through online courses and exercises.

Much effort has been made to achieve personalized learning in online-education [1, 38, 64]. Most of these methods are based on predicting students' knowledge states or recommending personalized learning resources using neural networks, based on sequences of student behaviors. However, this approach achieves only a coarse level of personalization. Even when students receive recommended resources, the specific difficulties they encounter in their learning process can remain unresolved. These difficulties may vary from student to student; for example, different students might struggle with different aspects of the same problem, such as misconceptions in understanding key concepts or difficulties in the reasoning process. These issues require detailed descriptions from each student for educators to understand. However, current online education systems face the issue of not being able to interact with students in real-time like a teacher. Presently, online education platforms typically offer static course videos and exercises, leaving students unable to ask questions or seek dynamic solutions to their specific issues. Therefore, developing a teaching assistant model capable of addressing students' individual concerns represents a crucial step forward in advancing online education.

The emergence of large language models (LLMs) instills optimism in creating an intelligent education system. Since the launch of ChatGPT, LLMs have shown exceptional ability in understanding human knowledge and have been widely applied in various professional fields, including recommendation systems [67], healthcare [73], economics [63], and others. By introducing expansive world knowledge, considerable reasoning capabilities, and the ability to understand human language, LLMs have the potential to introduce new forms of interactions and methodologies across these fields. However, the challenge intensifies when constructing an LLM-based education system. To solve the specific problems encountered by students, it's not only necessary for LLMs to understand the precise issues faced by the students but also essential for them to possess specialized educational-related knowledge and skills. Only then they would be able to address problems that the students themselves might not be able to resolve.

The act of LLMs answering students' questions can be considered as a process that involves using multiple educational capabilities. For instance, as depicted in Figure 1, the question posed by the student requires the education assistant to possess both mathematics and programming skills simultaneously. Therefore, summarizing the methods of developing LLMs' education-related capabilities is meaningful for building the next-generation intelligent education system. In this paper, we study LLMs from the perspective of these education-related capabilities and explore the potential of an LLM-based education system, aiming to provide valuable insights into the areas that need further improvement. We present our primary investigation in Figure 2. Our specific focus centers on five distinct capabilities of LLMs:

- **Mathematics:** We primarily summarize the current development of LLMs in solving mathematical problems of varying difficulties and types, encompassing basic arithmetic operations, challenging problems, multi-modal problems, and math proof.

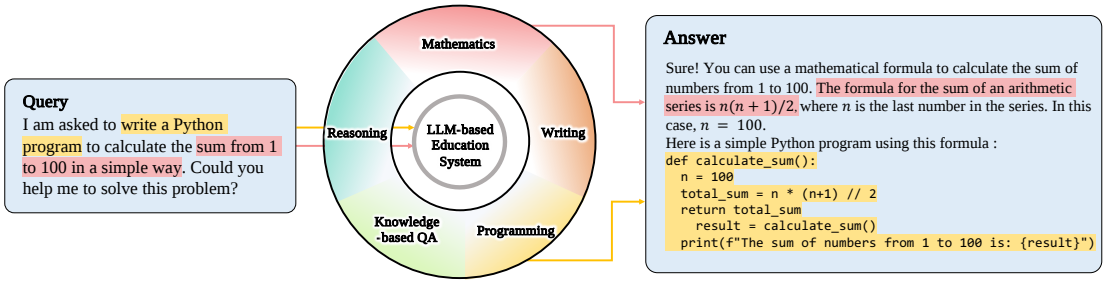


Fig. 1. An example of LLM-based education systems integrating multiple abilities to solve student problems.

- **Writing:** We investigate the performance of LLMs on a number of representative writing tasks to outline the problems LLMs face and potential future directions.
- **Programming:** In accordance with human programming conventions, we divide LLMs' programming process into two stages: code writing and code refinement. We review the researches in this area and summarize the remaining problems for LLMs in coding.
- **Reasoning:** We explore the capability of LLMs to perform reasoning in various ways, including supervised fine-tuning, prompt engineering, and hybrid methods, and explore their potential applications in the field of education.
- **Knowledge-based Question Answering:** We investigate the development of LLMs in open-domain and domain-specific knowledge-based question answering. We hope to offer a comprehensive view of incorporating such capabilities into the education system.

These five capabilities form the foundation of an LLM-based education system. Building upon this foundation, we propose two potential approaches for forming an LLM-based education system. One approach involves training a comprehensive language model with multiple capabilities, while the other is based on an LLM controller in a mixture-of-experts framework.

The rest of this paper is organized as follows. In section 2, we briefly introduce the education tasks and educational LLMs. In section 3, we summarize the current development status of five foundational abilities related to education, followed by discussion of the development trend of education-related LLM capabilities. In section 4, we investigate the performance of well-known LLMs in education-related capabilities. In section 5, we introduce the potential approaches for organizing an LLM-based education system. Finally, in section 6 and section 7, we highlight the challenges and future directions for designing an LLM-based education system and concludes this survey.

2 BACKGROUND

In this section, we first discuss the educational tasks and introduce of LLMs' roles in smart education. Then, we introduce the current development of educational large language models and compare our survey with previous works.

2.1 Educational Tasks

Artificial Intelligence can significantly boost the development of online education. Developing an intelligent education system involves tackling a wide array of tasks, which can be broadly categorized into two types. The first type centers around solving personalized, knowledge-based questions from students[148, 150, 160]. This category aims to assist

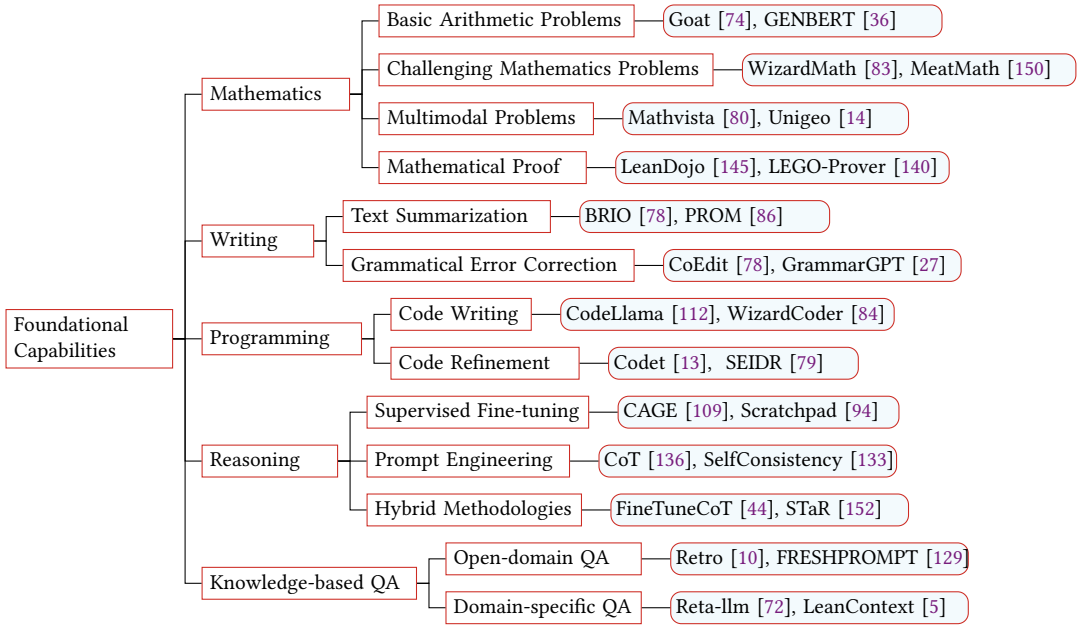


Fig. 2. Summary of LLM's education-related foundational capabilities.

students in resolving their queries during the learning process, such as clarifying misunderstandings about a particular concept[59], solving a math problem[39, 135, 141], or writing a piece of code to address a specific issue[13, 111, 158]. The second type focuses on aiding students with their learning planning, such as mapping out learning paths[18, 64], knowledge tracing[1, 20, 103], and computerized adaptive testing[37, 89, 124]. These tasks are designed to support the learning process from a broader perspective, instead of addressing specific learning challenges students face.

In this paper, we mainly survey the former category which mainly develops the abilities of LLMs answering students' specific questions requiring specific skills, due to that it's the main scenario that LLMs could contribute more. The former scenario is where LLMs currently play a significant role, for two main reasons: 1) From the perspective of task characteristics, tasks like designing learning paths and tracking knowledge, though also guiding student learning, are mainly based on students' learning sequences. The reasoning process mostly happens in the background, with relatively less need for dialogue. 2) From the viewpoint of the characteristics of LLMs, the advantage of language models over traditional recommendation models lies in their extensive world knowledge, conversational abilities, and logical reasoning capacities. These capabilities are crucial for addressing the personalized and subject-specific questions that students encounter. These questions are often complex and personalized, requiring dialogue for effective understanding—a feature not possessed by previous deep learning models. However, for issues like learning path planning and knowledge tracking, deep learning models, through training, can handle them well [37, 64, 103].

2.2 Educational Large Language Models

Currently, many online education companies have launched their own large educational models[41, 48]. The iFLYTEK Spark[48], introduced by iFLYTEK Company, boasts capabilities in multimodal interaction, coding, text generation, solving math problems, and knowledge Q&A. The introduction of LLMs could evidently enhance the efficiency of

learners [57]. MathGPT[41], developed by TAL Education Group, is an LLM specializing in math problem-solving and lecturing. From the development patterns of these companies, it is evident that the primary application scenario for these large models is to address specific knowledge-based questions from students. This approach benefits the industry in two main ways: 1) It allows for more interaction with students. Answering specific questions about subjects or knowledge points involves more detailed participation in the students' learning process compared to learning path planning and knowledge tracking, thereby increasing the time students spend on the platform. 2) Solving student questions makes the model appear more intelligent. Planning learning paths and assessing students can be achieved with traditional models, but understanding and solving students' problems through dialogue is only possible with large models, making the platform's products more intelligent. In summary, because they better retain student users and make their products more intelligent, the main trend for platforms developing educational large models is to enable them to solve specific student questions. Of course, issues like learning path planning and knowledge tracking are also very important, but the transformation in the era of large models requires further exploration.

2.3 Related Surveys

LLMs hold vast potential in the field of education. There are already several surveys about LLMs in education, while our work is different from them. Gan et al. [34] explored various roles LLMs can play in the education process. It focuses on analyzing the roles undertaken by LLMs from the perspective of different application scenarios, such as learning support tools, personalized learning experiences, content creation and generation, language learning and teaching, cross-lingual communication, and translation. The basic capabilities of LLMs are not discussed in it. Kasneci et al. [56] explores the benefits and challenges of LLMs in education from both students' and teachers' perspectives, highlighting the potential of LLMs in research, writing, and problem-solving tasks, as well as their ability to provide domain-specific language skills for specialized learning.

Other than that, AL-Smadi [4] primarily explored the use of generative AI models in education, focusing on their application as teaching aids, the generation of personalized learning materials, and the assessment of student learning outcomes. It primarily assesses ChatGPT's performance on tasks such as instructional design from an educational perspective, in a qualitative rather than quantitative manner. Meyer et al. [90] is an editorial on the opportunities and challenges of LLMs in academia, analyzing the potential impacts and risks of LLMs from the perspectives of academic writing, education, and programming education. They primarily discussed the role of LLMs in education from a pedagogical perspective, whereas our work leans towards analyzing the current capabilities of LLMs in solving subject-specific questions in the educational process from a technological standpoint, offering ideas and insights for creating an LLM capable of addressing problems across various subjects. Wang et al. [131] summarized the development of LLMs in education from the perspective of data and technology, discussing the current applications of LLMs in tasks such as study assisting, teach assisting, and adaptive learning. It is important to note that while the article mentions some technical methods for developing LLMs' problem-solving abilities in study assisting, the content covered is not systematic. The technical development for solving problems for students didn't take up much space of this article. Moreover, the disciplines explored in the article are not foundational but include advanced subjects such as Medicine and Finance. In contrast, our work primarily focuses on discussing the development of LLMs' fundamental capabilities in assisting students with solving various problems.

While previous surveys have provided ample discussion on the potential applications of LLMs in education, they exhibit two main shortcomings: 1) Their exploration of LLMs' applications in education often spans a wide range of topics, including designing learning paths, assisting teachers, and planning curricula. As discussed in Section 2.1, although these

capabilities are important, they do not represent the primary direction of current practical applications of educational LLMs. Our work, in contrast, primarily focuses on the ability of LLMs to answer subject-specific questions. 2) They have not analyzed the development of LLMs' educational capabilities from a technological perspective. Discussing the evolution of LLMs' educational abilities from a technical standpoint is crucial for building a general LLM-based intelligent education system. Different from them, we review the evolution of LLMs from the perspective of education-related capabilities. We summarize the techniques to promote LLMs in these capabilities. Additionally, we provide foresight into constructing feasible frameworks for LLM-based education systems. Our work emphasizes a comprehensive understanding of LLMs' educational competencies and explores frameworks that can effectively integrate these abilities into the educational landscape.

3 FOUNDATIONAL CAPABILITIES

3.1 Mathematics

Mathematics demands reasoning of complex information, making it one of the disciplines that place the highest premium on the cognitive abilities. There is significant academic interest in enhancing the mathematical capabilities of LLMs [82]. In the pursuit of creating an education system based on LLMs, the goal is to equip it with the capability to tackle a wide range of mathematical problems. These problems may encompass basic numerical calculations, complex logical reasoning, or challenges that require the integration of information from multiple modalities. In this section, we summarize the developments of the mathematical capabilities of LLMs, focusing on four aspects: fundamental numerical computations, complex reasoning, the handling of multi-modal problem-solving, and mathematical proof.

3.1.1 Basic Arithmetic Problems. Recently, significant scholarly attention has focused on augmenting LLMs' proficiency in this domain [36, 99, 135, 154]. In human learning mathematics, foundational mathematical operations serve as the basis for addressing more advanced problems. Given the robust comprehension of human language and notable textual reasoning abilities exhibited by LLMs, it is natural for many to assume that LLMs should effortlessly handle fundamental mathematical problems. However, the reality proves otherwise. Yuan et al. [151] pointed out that ChatGPT and GPT-4 [2] perform well in addition and subtraction operations, but their accuracy decreases when dealing with multiplication involving larger numerical values. This limitation arises because the LLMs do not access a calculator during the computations. More importantly, when LLMs solve computational problems, their internal logic does not perform actual calculations. Instead, they generate text to predict each digit of the answer step by step. This approach means that as the number of digits in the answer increases, the probability of making an error grows cumulatively.

This problem is not unsolvable. Yang et al. [146] and Liu and Low [74] proposed to fine-tune LLMs on high-quality datasets and found that even small language models could avoid making mistakes on multi-digit problems. Lee et al. [60] found that even small transformers could solve arithmetic problems with high accuracy as long as trained on data with detailed calculation process. All in all, fine-tuning on high quality data is one workable solution. However, this approach may only work when a specialized arithmetic model need to be build while not applicable for building general LLMs since it is not feasible to fine-tune each LLM individually. How to prevent such issues during the pre-training phase of LLMs remains an open question. At the current stage, a simple and feasible solution is to have the LLMs outsource arithmetic problems to a calculator, which can ensure the accuracy of the calculations [115].

3.1.2 Challenging Mathematics Problems. Despite occasional errors in basic mathematical operations, the expectation for LLMs to solve more complex problems remains high, and the LLMs' capabilities in this area continue to develop.

For education, the ability to handle college-level mathematics is particularly beneficial for senior students' learning, offering significant assistance in understanding challenging concepts. For simple arithmetic problems, LLMs make mistakes mainly due to the gap between text generation and digit calculation. For complex mathematical problems, the challenge arises for requiring LLMs' advanced symbolic reasoning abilities and domain knowledge. In this regard, LLMs still need further development. Wang et al. [132] introduced a benchmark SCIBENCH, which contains collegiate-level scientific problems from mathematics, chemistry, and physics textbooks, while GPT-4 could only get averagely 53.24% on the math part. Furthermore, Sawada et al. [114] collected a harder dataset ARB that contains math problems from Harvard PhD comprehensive exams in mathematics, where GPT-4 could only get less than 10% right. All these results demonstrate that LLMs have a lot of room for improvement.

In recent years, more and more work has been proposed to enhance this ability [62, 83, 132, 150]. Luo et al. [83] tried to increase the mathematical reasoning abilities of Llama-2 by applying Reinforcement Learning from Evol-Instruct Feedback (RLEIF) on complex mathematics datasets. The Evol-Instruct method made LLMs to generate easier and harder questions from the original questions to make the LLMs think deeper. In addition to training on high-quality datasets, there are also many efforts that attempt to leverage programming as an external tool to assist in solving mathematical problems. Zhou et al. [160] tried to enhance GPT-4's mathematical ability by encouraging it to use code to self-verify its answers. This approach leads to a significant improvement in the zero-shot accuracy of mathematical problem-solving. ToRA [39] divided the process of LLMs solving math problems into a cyclical rationale-action process, where the action involves invoking external tools, including computation libraries and symbolic solvers, thereby amalgamating the analytical prowess of language with the computational efficiency of tools. Overall, solving complex mathematical problems requires reasoning abilities, computational power, and knowledge in the mathematical domain. Fine-tuning on relevant datasets primarily aims to enhance its reasoning capabilities and knowledge on related issues, while computational abilities can be supplemented by invoking external tools, such as calculators or code compilers.

3.1.3 Problems Involving Multi-Modal Information. Multi-modal inputs are common in mathematics problems like geometric questions. They require LLMs to understand text and image information for solutions. Research in multi-modal LLMs for mathematical reasoning is emerging [14, 80, 101]. This type of task poses a high requirement for the formation and quality of the training data. Chen et al. [14] introduced a Unified Geometry problem benchmark combining calculation and proving tasks. Based on this dataset, the study presented a framework capable of simultaneously solving calculations and proving tasks through a sequence generation approach. Furthermore, Lu et al. [80] proposed MATHVISTA, a benchmark for diverse mathematical and visual challenges. Zhang et al. [155] developed a benchmark GeoEval for testing geometry problem-solving ability of LLMs. Their results showed that WizardMath and GPT-4V excels in handling multi-modal mathematics problems.

Various methods have been proposed for this task. For geometric problems, Zhang et al. [156] converted diagrams into text clauses, using a convolutional neural network and a language model for encoding and a GRU-based framework for answer generation. Gao et al. [35] argued that the reason current models fail on solving geometry problems is that they struggle to accurately comprehending basic geometric elements and their relationships. So they built a augmented dataset Geo170K containing high-quality descriptions of geometric information and developed a model G-LLaVA on it, which demonstrated exceptional performance in solving geometric problems, significantly outperforming GPT-4-V on the MathVista benchmark with only 7B parameters.

Besides geometry problems that involves processing images and text, Lu et al. [81] presented the Tabular Math Word Problems (TABWP) dataset, requiring textual and tabular data reasoning, and introduced PROMPTPG, a policy gradient-based selector for training and prompt construction for test samples.

3.1.4 Mathematical Proof. Unlike other types of mathematical problems where LLMs primarily focus on providing answers, LLMs' role in mathematical proofs emphasizes the integration with proof assistants such as Coq [8], Isabelle [93], and Lean [22]. These proof assistants correspond to specific programming languages, requiring users to formulate proofs in the required languages, after which the assistant can verify the proof's correctness. Many LLM-based methods are proposed to help theorem proving [50, 71, 141]. Based on these proof assistants, there are two main approaches to utilizing LLMs for mathematical proofs.

The first approach is formal proof search, exemplified by models like GPT-f [104], which involves prompting LLMs to produce the next proof step (also called 'tactic' in proof assistants) based on the current proof state and some optional context. In conjunction with proof assistants, it transforms the proof of mathematical propositions into a process of executing actions. Here, an action can be the application of a mathematical theorem or a method of variable substitution, which can transform and decompose the original proposition. The LLM is responsible for generating actions, sampling multiple actions in each round and iterating over multiple rounds to produce a tree structure. It utilizes the proof assistant's functionality to verify the validity of proofs to evaluate branches, thereby employing tree search methods to find proof strategies. Following GPT-f, Thor [51] was further proposed to help select the premise for theorem proving. Yang et al. [145] introduced an open-source framework named LeanDojo based on the Lean proof assistant. This framework comprises data, toolkits, models, and benchmarks, and it has led to the development of ReProver (Retrieval-Augmented Prover), which enhanced proof accuracy by using retrieval methods to extract premises for LLMs to base their mathematical proofs.

The second is natural proof translation, also known as autoformalization, which is to convert math proofs written in natural language into formalized versions. In such schemes, the responsibility of LLMs is not to generate proof steps. Due to the extremely low prevalence of these proof assistants in the human corpus, LLMs struggle to directly undertake the task of generating proof steps. This approach primarily tackles the challenge of insufficient data for formal mathematical proofs. By leveraging autoformalization, a significant increase in this type of data can be achieved, consequently enhancing the proof-generating capabilities of neural provers that have been fine-tuned on this expanded dataset. Initially, Wu et al. [138] demonstrated that LLMs perform well in autoformalization. They employed LLMs for autoformalization, transforming mathematical proofs and problems expressed in natural language into formal specifications and proofs in Isabelle language. The generated data was used to train a neural theorem prover, enhancing the effectiveness of the original prover. Following it, Cunningham et al. [21] utilized an encoder-decoder framework based on the universal transformer architecture, converting both problem statements and mathematical proofs written in LaTeX into the language of the Coq interactive prover. Jiang et al. [52] built a pipeline of Draft, Sketch, Prove (DSP), where the informal and incomplete proof is first generated (Draft) and given to LLMs for autoformalization (Sketch), and finally passed to off-the-shelf prover to be completed (Prove).

In mathematical education, proof problems are indispensable. Currently, LLMs for math proof primarily operate in the form of interactive theorem proving. In this approach, LLMs complete proofs by interacting with software proof assistants. To realize completely automated theorem proving with LLMs, it is essential that these models possess not only strong reasoning skills but also the capability to formalize concepts effectively. There is no room for hallucination in mathematical proofs, which poses a formidable challenge for LLMs.

3.1.5 Summary. When examining the progress of large language models (LLMs) in terms of their mathematical abilities, it becomes evident that the primary obstacle lies in the inherent conflict between the principles of mathematical logic and those of text generation. This discrepancy manifests itself not only in the outcomes (e.g., LLMs encountering difficulties with multiplication involving large numbers and struggling with complex mathematical problems) but also in the training data itself. Mathematical problems, in their symbolic form, represent only a minor fraction of the training corpora used for these expansive models. Consequently, the current approaches to bolstering the mathematical capabilities of LLMs can be broadly categorized into two main strategies: 1) Data enhancement: The most straightforward method to improve LLMs' performance on mathematical tasks is to provide them with high-quality, relevant data during the fine-tuning phase of their training. By exposing the models to a more comprehensive and representative set of mathematical problems, their ability to handle such challenges can be significantly enhanced. 2) Tool integration: Another effective approach is to leverage external tools, such as calculators and code compilers, to compensate for the inherent limitations of LLMs. By strategically invoking these tools at the points where the models struggle, their functional shortcomings can be effectively mitigated, allowing for a more comprehensive and accurate handling of mathematical problems.

3.2 Writing

Writing proficiency is crucial for LLMs, underpinning their ability to comprehend inputs deeply and produce semantically and syntactically accurate outputs [12, 25]. In education, the writing capability of LLMs holds the potential to transform how writing is taught. They can assist in content creation, simplify complex topics for students, and offer personalized educational materials. In this part, we dive into LLM's writing capability on two education-related tasks: text summarization and grammatical error correction.

3.2.1 Text Summarization. Text summarization is a task that requires LLMs to compress lengthy texts into concise summaries while maintaining the essential information. This process presents a significant challenge for LLMs, as they must effectively comprehend and distill the key points from a wide range of diverse content, such as news articles and texts written in multiple languages. In the context of education, students are often confronted with an overwhelming volume of intricate learning materials. A well-crafted summary can prove invaluable in helping them grasp the core concepts quickly and efficiently, saving them considerable time and effort. For instance, a summary can break down a complex piece of code into its fundamental components, making it easier for students to understand its structure and functionality. Similarly, a summary can highlight the main ideas and key takeaways from a lengthy chapter, allowing students to focus on the most critical information without getting lost in the details. It is evident that traditional fine-tuning methods are less effective with the advent of advanced LLMs [78, 105]. Pu et al. [105] and Liang et al. [65] showed LLMs like ChatGPT initially lag behind fine-tuned models like T5 [107] and BART [61] in ROUGE scores [66] for text summarization. However, when human judges evaluate overall quality, LLMs outperform fine-tuned models and even standard human summaries, superior in aspects like factual consistency, fluency, and diversity. This discovery underscored the limitations of traditional evaluation methods and suggested a need for new paradigms to guide summarization tasks in the LLM era. For example, BRIO [78] implemented a ranking task to foster more diverse summarizations. Furthermore, Liu et al. [77] utilized a GPT model based on BRIO to directly generate training data to guide the learning process of other models, which is similar to the process of RLHF [120].

Given the outstanding performance of LLMs in the domain of text summarization, researchers have already begun to tackle more challenging tasks. Liu et al. [76] benchmarked LLMs on instruction controllable text summarization. In

this task, the input provided to the model consists of two components: the source article that needs to be summarized and a set of instructions in natural language that specify the desired characteristics of the summary output. The goal is to evaluate how well LLMs can generate summaries that adhere to these specific requirements. In a related study, Shen et al. [116] investigated whether LLMs could potentially replace human evaluators in assessing the quality of abstractive summarization. Abstractive summarization involves generating a summary that captures the main ideas of the source text while potentially using different words and phrases. The researchers found that, at present, LLMs are not capable of serving as reliable substitutes for human evaluators in this task. LLM evaluators rate each candidate system inconsistently and are dimension-dependent. Moreover, LLMs face challenges when comparing candidate summaries that have similar levels of performance. They find it difficult to make fine-grained distinctions between summaries of comparable quality, which limits their ability to provide accurate comparative assessments. The correlation between the ratings provided by LLMs and those given by human evaluators becomes lower when dealing with higher-quality summaries. Although LLMs can surpass humans in text summarization tasks, they are not without flaws. Current LLMs make fewer silly mistakes (e.g., entity confusion, irrelevant information generation) but more sophisticated ones [105]. For example, they fill in the details related to but not directly supported by the source text, which is a kind of "hallucination". Liu et al. [75] tried to employ human feedback to enhance the summarization factual consistency. The dataset DeFacto they built contained human demonstrations and informational natural language feedback consisting of corrective instructions, edited summaries, and explanations with respect to the factual consistency of the summary. Feng et al. [29] tried to resolve this hallucination problem by disentangling the comprehension and embellishment abilities of LLMs. It trained the embellishment to be consistent with the facts presented in the original text.

Overall, LLMs perform well in text summarization tasks, even surpassing humans in simple summaries, but this does not mean they are flawless. In the educational domain, helping students summarize learning materials should ensure there is no conflict between the summary and the original content. LLMs still face issues with hallucinations in this aspect. While these hallucination issues can be mitigated through post-processing techniques, hallucinations remain a fundamental problem with LLMs that extends beyond the task of text summarization. Addressing the issue of hallucinations in LLMs is an ongoing research challenge that requires further investigation and development of novel approaches. Until a satisfactory solution is found, it is important to exercise caution when using LLM-generated summaries in educational contexts and to have mechanisms in place to verify the accuracy and consistency of the summaries with the original learning materials.

3.2.2 Grammatical Error Correction. We are well aware of the remarkable capability of LLMs to generate fluent and coherent conversations. However, from an educational perspective, the importance of producing grammatically correct dialogues cannot be overstated, especially for students learning a new language. The correctness of grammar in conversations plays a vital role in language acquisition, providing students with reliable examples to emulate and learn from. Numerous studies have evaluated the effectiveness of LLMs in grammatical error correction (GEC). Several works [28, 91, 137] first evaluated the error correction performance of closed-source LLMs such as ChatGPT. Although there exists a pronounced gap between ChatGPT and the previous state-of-the-art [40, 95] models on the overall F0.5 metric, closer analysis shows that ChatGPT underperforms other models in terms of precision but far exceeds other models in terms of recall. That said, LLMs like ChatGPT are good at error detection. A detailed manual analysis of ChatGPT's outputs revealed that, in most cases, it maintained grammar accuracy better than the previous methods. However, it often overcorrects sentences to increase diversity and fluency, resulting in a decrease in the recall score. As a result of this characteristic, LLMs perform better when evaluated on higher-order metrics such as fluency, which assess

modifications to text. However, for issues requiring minimal edit corrections, they may not necessarily outperform traditional models. Some efforts have attempted to mitigate the problem of over-correction in large models by employing instruction-tuning techniques, encouraging them to make only the necessary, such as CoEdit [108], which covered multiple text editing tasks (including GEC) by fine-tuning LLMs to integrate the capabilities brought by these tasks. GrammarGPT [27] collected grammatically incorrect sentences and performed instruction tuning on LLMs to improve the ability of positioning grammar errors.

Overall, LLMs perform well in the area of GEC, with their main issue being over-correction. In scenarios such as copywriting or article writing, this problem is not critical, as LLMs can assist individuals in correcting grammatical errors while crafting more fluent sentences. However, in educational settings, LLMs' GEC capabilities are more often utilized to aid students in learning grammar. This requires LLMs to accurately identify grammatical errors in sentences. The issue of over-correction could potentially mislead students, making further adjustments necessary.

3.2.3 Summary. Leveraging LLMs' proficiency in text summarization and grammatical error correction can significantly benefit education. Their capability to condense complex material into concise summaries facilitates efficient learning, while error correction tools help improve students' writing and language skills. However, critical challenges need to be resolved to integrate these writing-related capabilities to help education. It becomes evident that more refined evaluation metrics and task-specific optimizations are essential for LLMs.

3.3 Programming

Programming is a process of writing code and correcting code if unexpected results are obtained. Incorporating LLMs in programming education is reshaping the future of AI-assisted programming learning. LLMs could play multiple roles: as instructors providing guidance, as teaching assistants offering personalized tutorials, and as collaborative coding partners. Studies like [85] demonstrated improved performance (17%) and efficiency (13%) among programming novices using LLM-based assistants. Research from [102] focused on programming education tasks and benchmarks like [32] and [24], which were used to evaluate the effectiveness of LLMs. This section mainly discusses the LLMs' coding capability development from two perspectives: code writing and code refinement, corresponding to the two stages in programming.

3.3.1 Code Writing. Unlike natural language tasks, generating code requires a more rigorous token syntax and places higher demands on the training stage. A common method to improve LLMs' performance in generating code is to train or fine-tune them on extensive code datasets [15, 92]. WizardCoder [84] introduced the Evol-Instruct [142] method to generate complex and diverse instruction datasets of code-related tasks. To emulate the iterative process of humans repeatedly modifying and reviewing code, InCoder [31] utilized bidirectional encoding instead of left-to-right encoding. In addition to next-token prediction, training or fine-tuning code-aimed LLMs on additional code-related tasks could enhance their programming capabilities. LLMs first learn language patterns and representations from a large amount of text data through unsupervised learning. Then, they could be fine-tuned on labeled code tasks, allowing them to learn targeted code representations and gain a deep understanding of code structure and semantics based on the provided labels. CodeT5+ [134] introduced the concepts of unimodal and bimodal alignment, increasing the model's adaptability to function in different modes for various downstream tasks. During the bimodal alignment phase, the model synchronizes the representations of text-code pairs using multiple tasks, which improves its ability to understand and generate content across different modalities. CodeLlama [112] also applied the multi-task objectives, including autoregression and causal infilling prediction, which achieves better performance among open models. MFTCoder

[70] utilized the Multi-Task Learning (MTL) technique and incorporated a training loss computation algorithm to alleviate the instability and imbalance of multi-task training. Given that code text has its unique syntax and structure compared to natural language text, the methods mentioned above all attempt to construct datasets of code and perform fine-tuning. This is the most direct and effective approach to enhancing the code capabilities of LLMs.

By fine-tuning on code datasets, LLMs can enhance the probability of generating a correct piece of code. Considering the human programming process, besides relying on the programmer's coding skills, it also involves various stages such as consulting documentation, designing code frameworks, implementing and testing submodules, which entail numerous decision-making processes. Therefore, many works perceive LLMs as agents, viewing the coding process as a continuous series of decisions and external tool invocations. Zhang et al. [158] attempts to enhance the effectiveness of model code generation by employing a tree search approach, without altering the parameters of LLM itself. Specifically, each token in the code is regarded as an action, with the generated code serving as the state. The LLM makes decisions step by step, while utilizing Monte Carlo Tree Search (MCTS) to calculate the value of each action (token) in the current state, thereby selecting the optimal action and significantly improving its pass rate in code generation. Similarly, Zhou et al. [161] also treats LLM as an agent, where each action involves generating a complete piece of code. It employs MCTS to estimate the value of each action as well. Shinn et al. [118] introduced Reflexion framework, which enhances LLM agents through linguistic feedback. The method assigned LLMs the roles of both generating code as actors and evaluating it the code as evaluators. Additionally, it utilizes self-reflection to generate verbal reinforcement cues aimed at assisting the actor in self-improvement. [163] introduces a document retriever as a precursor to the code generator, extracting relevant function descriptions from documents to provide external information to the LLM. This enables the generated code to utilize the latest library functions. By considering the LLM as an agent and utilizing external documents or tree search algorithms, the accuracy of the agent's decisions can be improved without the need to update the model's parameters, which reduces training costs. However, this approach also has a downside: it increases the time required for decision-making during the code generation process, resulting in lower inference efficiency compared to using the LLM alone.

In addition to single-agent approaches, multi-agent systems have also made significant progress in code generation tasks. Qian et al. [106] developed ChatDev that divided the process of writing code into four stages: designing, coding, testing, and documenting. Each stage is managed by a group of "software agents", with the entire chat chain acting as a facilitator, assigning specific sub-tasks to each stage. The system achieved efficient code writing. Furthermore, Hong et al. [45] proposed MetaGPT, a Multi-agent system for coding, which assigns diverse roles to different agents, such as Product Managers, Architects, Engineers, etc. They introduce Standardized Operating Procedures (SOPs) into prompt sequences, effectively enhancing the effectiveness of code generation. Although chat-based multi-agent systems have shown significant effectiveness in code generation tasks, they impose high demands on the fundamental capabilities of LLMs due to the need for dialogue-based coordination between agents. The effectiveness of these systems generally increases with the rise in the parameter size and capability of LLMs.

3.3.2 Code Refinement. In most cases, LLMs could not generate the correct code at once. We could enable LLMs to generate a code sketch (either actual code or pseudocode) and utilize various methods to guide the model to modify and refine the code. By leveraging the inherent code correction ability of LLMs, the overall precision and quality of the code could be significantly enhanced.

We investigate the development of LLMs in code refinement from two perspectives. One aspect is the advancement of LLMs in the task of fixing code bugs, known as Automated Program Repair (APR). Sobania et al. [119] conducted

experiments analyzing the performance of ChatGPT on the APR task. They found that it can achieve competitive results compared to previous deep learning-based methods, and incorporating additional information through dialogue can surpass previous approaches. Xia and Zhang [139] introduced conversational APR, enabling LLMs to obtain bug feedback through dialogue, effectively enhancing the performance of various LLMs in APR. In addition to the APR task itself, the second aspect involves integrating code refinement into the process of code generation, leveraging bug feedback to enhance the effectiveness of code generation. Liventsev et al. [79] constructed a pipeline: Synthesize, Execute, Instruct, Debug, and Rank (SEIDR). It first generates multiple different codes and undergoes the process of code filtering and debugging, ultimately selecting the best code among them. According to Magister et al. [87], teaching an LLM to debug its program draft via few-shot demonstrations could improve the performance on code generation tasks. Another method for LLM debug is generating unit tests by LLM itself and checking its code [13]. By mimicking the human coding process, LLM's programming ability is greatly enhanced. However, these methods lead to an increased number of calls to LLMs, resulting in a significant increase in inference time.

In the context of coding education, the support and guidance provided by LLMs do not yet match the level of assistance offered by human instructors. One major reason for this gap is that LLMs still have substantial room for improvement in their coding capabilities. While LLMs can generate functional code for relatively simple tasks, when it comes to generating complex algorithms, LLMs' performance rapidly declines compared to the functionality achieved by humans [16]. Additionally, due to the lack of real-world data, LLMs struggle to learn the intermediate thinking process of code writing, making it difficult for them to provide relevant explanations and instructions to beginners. As a result, the use of LLMs in programming education still needs to be improved, especially in terms of interpretability.

3.3.3 Summary. Code data is more abundant in the training corpora of LLMs compared to mathematical data. This is primarily due to the inherent nature of coding, which heavily relies on computers and the internet. As LLMs are trained using data scraped from the web, they are exposed to a significant amount of code-related information during the training process. Many improvements to LLMs' code generation abilities are inspired by the human programming process. For instance, programmers often refer to documentation and resources to gain a better understanding of the problem and potential solutions. For complex coding tasks, the solving process typically involves a cycle of design, writing, and debugging. These thought processes can be utilized to enhance the programming effectiveness of LLMs. From an educational perspective, it is crucial for LLMs to not only generate correct code but also possess the ability to analyze and provide feedback on code written by students. This involves identifying issues, suggesting improvements, and offering explanations to help students learn and grow as programmers.

3.4 Reasoning

The reasoning capability of LLMs offers significant potential for educational use, serving as advanced tools that enhance students' cognitive processes, provide personalized mentorship, and offer tailored learning support. This section reviews LLMs' general reasoning ability development strategies.

3.4.1 Supervised Fine-tuning for Reasoning. Previous studies have primarily focused on fully supervised fine-tuning LLMs to enhance their reasoning capabilities. This approach aligns model outputs closely with labeled datasets, allowing the models to produce highly accurate predictions within specific domains. One such study by [109] demonstrated the efficacy of fine-tuning a pre-trained GPT model, which generated rationales for predictions on the CoS-E dataset [121]. The results revealed that models trained with explanations exhibited improved performance in commonsense question-answering tasks. However, the effectiveness of fine-tuning methods heavily relies on the availability of a

specific dataset that includes explicit reasoning steps. Acquiring such a dataset can prove to be challenging. Moreover, the scope of inference from fine-tuned models is restricted to the dataset's domain, hinging largely on the data's inferential quality. This constraint highlights the benefits and limitations of fully supervised fine-tuning, as it narrows the model's reasoning abilities to the dataset's specific domain. Consequently, it underscores the need to explore methods that harness LLMs' intrinsic reasoning capabilities, potentially providing broader relevance and deeper insights beyond the limitations of domain-specific datasets.

3.4.2 Prompt Engineering for Reasoning. Efforts have been made in recent research to tackle the constraints inherent in the fine-tuning process of LLMs. These fine-tuning methods tend to overfit specific dataset distributions, reducing their effectiveness on more diverse datasets. In response to this issue, a variety of strategies have been proposed. These strategies are designed to draw upon the robust reasoning abilities inherent in LLMs by leveraging their extensively pre-trained parameters. One approach involves guiding LLMs to generate inference and reasoning through demonstrations or prompts. For example, Wei et al. [136] introduced the "Chain of Thought" (CoT) method, which utilized natural language reasoning steps as prompts for the model. By integrating CoT within a few-shot prompting framework, the model leveraged its extensive parameters to produce analogous chains of reasoning. Consequently, this approach empowered the model to adeptly navigate complex reasoning tasks across diverse domains, obviating additional training or fine-tuning. This innovation underscored the model's inherent capability to generate deductive pathways, significantly enhancing its applicability and versatility in problem-solving scenarios without extensive domain-specific adaptations. Similarly, Wang et al. [133] introduced a self-consistency strategy that enhances model performance by sampling various reasoning paths and selecting the most consistent answer. This approach diversified the exploration of reasoning strategies. It ensured the conclusions' reliability, showcasing an innovative way to leverage the model's capabilities for improved decision-making and problem-solving across different contexts. Confronting the limitations of relying on static, manually annotated demonstrations, which can restrict the adaptability of LLMs to the varying complexities of real-world tasks, Diao et al. [23] introduced an active selection approach. This technique dynamically pinpointed the most pertinent demonstrations aligned with the specific demands of a task from a broad set of queries. In this way, the approach enhanced the flexibility and effectiveness of LLMs in adapting to diverse and evolving problem contexts. Concurrently, Zhou et al. [162] devised a prompting methodology that broke down intricate problems into their simpler constituent sub-problems. This tactic not only promoted a step-by-step problem-solving process but also hold promise for augmenting the efficacy of LLMs in handling complex tasks.

Building upon the CoT methodology, subsequent developments have introduced more sophisticated frameworks for enhancing the reasoning capabilities of LLMs. The Tree of Thoughts (ToT) [147] framework extended CoT by enabling LLMs to explore multiple reasoning paths through a hierarchical structure, thereby improving decision-making for tasks requiring strategic planning. Following the ToT, the Boosting of Thoughts (BoT) [17] framework introduced a novel approach by iteratively exploring and self-evaluating multiple trees of thoughts. This process accumulated an ensemble of trial-and-error reasoning experiences, offering a new form of prompting designed to tackle complex problems. Starting with simple prompts, BOT iteratively refined reasoning steps through error analysis, significantly improving the generation of reasoning paths and achieving higher problem-solving rates on complex tasks than existing advanced prompting strategies. To structure thoughts through prompts without depending on fine-tuning, the Graph of Thoughts (GoT) framework [9] introduced a new angle by arranging thoughts generated by LLMs into a graph structure. This setup fostered a dynamic interaction among different thought units, facilitating synthesizing synergistic results, simplifying intricate thought networks, and refining ideas via feedback mechanisms. GoT's graph-based approach

presented a versatile tool for problem-solving, allowing for a more nuanced and interconnected reasoning process that mirrors the complexity of human thought.

The emergence of CoT and related prompting strategies marked a significant evolution in utilizing LLMs for advanced reasoning, transitioning away from dependence on fine-tuning. These methods exploited the LLMs' inherent capabilities to enhance their flexibility and effectiveness across various tasks without reliance on domain-specific tuning.

3.4.3 Hybrid Methodologies for Reasoning. Despite the success of prompt engineering in leveraging the intrinsic characteristics and capabilities of LLMs to enhance their performance, this approach falls short of fundamentally augmenting the model's core reasoning abilities, as it does not alter the model's underlying parameters. This inherent limitation points to a need for strategies that not only exploit the pre-existing strengths of LLMs but also seek to expand their innate capabilities. Innovative approaches that integrate the specificity of fine-tuning with the flexibility of prompt engineering have been developed to bridge this gap. These hybrid methodologies aim to bolster the LLMs' responsiveness to complex prompts and substantially improve their intrinsic reasoning capacities, offering a more comprehensive enhancement of their problem-solving prowess. One practical approach is to utilize LLMs to "teach" language models with smaller model sizes. Ho et al. [44], Magister et al. [87] explored to fine-tune a student model on the chain of thought outputs generated by a larger teacher model and proved that enriching the fine-tuning data with such diverse reasoning results in a substantial performance boost across datasets even for very small models. Moreover, Zelikman et al. [152] reported significant performance improvements across multiple datasets by generating step-by-step rationales and fine-tuning models based on correct answers, thus facilitating model learning from its reasoning. Similarly, Huang et al. [47] proposed that by employing chain-of-thought prompting [136] and self-consistency [133] to generate rationale-augmented answers, and then used these answers for fine-tuning, LLMs autonomously refined their reasoning capabilities. This approach highlighted the significant ability of LLMs to advance their knowledge and problem-solving skills independently.

3.4.4 Summary. In the educational domain, reasoning tasks possess unique characteristics that necessitate not only the accurate processing of information but also the capability to navigate and elucidate complex concepts in a manner that is accessible and educational for learners. As mentioned above, the discussed methods have significantly advanced the reasoning capability of LLMs, optimally utilizing their unique features for diverse reasoning tasks. This enhancement can greatly benefit educational applications. However, it's crucial to recognize the limitations. As underscored by Valmeekam et al. [128] and Ruis et al. [113], LLMs struggle with complex reasoning tasks and those requiring implicit expressions. For example, LLMs can struggle with complex reasoning scenarios, leading to a notable decrease in performance. This is particularly relevant in educational contexts, where incorrect problem-solving modeled by LLMs could misguide students and lead to misunderstandings or flawed comprehension. Thus, despite LLMs' immense educational potential, their limitations must be carefully considered to ensure they facilitate rather than obstruct learning.

3.5 Knowledge-based Question Answering

In the context of Knowledge-based Question Answering using LLM, the user presents a question to LLM, and LLM leverages knowledge-based methods and responds with the corresponding answer. Previous work by [110] showed that LLMs have an inaccurate perception of factual boundaries and often exhibit overconfidence. Many studies have explored and utilized external knowledge from open-world and domain-specific databases to enhance the knowledge base of these LLMs.

3.5.1 Open-domain QA. Open-domain question answering requires LLMs to accurately determine the reliability of information in the open world and craft their responses based on that understanding. The critical requirements for open-domain QA are real-time responsiveness and authenticity. LLMs exhibit disadvantages in both of these aspects. Due to the fixed parameters of the model, ensuring real-time information solely through the LLM itself is challenging, and LLMs commonly suffer from severe hallucination issues [143], posing a challenge to their authenticity as well. Jiang et al. [54] evaluated the accuracy of LLM responses to a particular question from the perspective of calibration. Through experiments, the researchers discovered that models such as T5, BERT, and GPT-2 are not well-calibrated in QA tasks. While suggesting that incorporating calibration-related methods into the fine-tuning process can effectively enhance performance in QA tasks, it is evident that solely pre-trained language models still face significant challenges in open-domain tasks. To overcome this challenge, many works tried to add additional information to help the LLMs answer correctly [10, 43, 58]. A common information source used is from the web. Lazaridou et al. [59] employed information gathered from web searches as prompt input for LLMs, conditioning it to generate answers to questions. This approach effectively enables LLMs to use open-world information to answer questions. Vu et al. [129] introduced FreshPrompt, which incorporated web pages collected from the internet into prompts given to large-scale models. This allowed them to leverage the latest information when answering questions. Kasai et al. [55] developed a QA platform REALTIME QA that updated itself weekly. Through evaluation on this platform, they found that GPT-3 could update its generation results based on newly retrieved documents. However, when retrieved documents fail to provide sufficient information to find the answer, GPT-3 may provide outdated answers.

The development of LLM-based open-domain question answering highlights significant challenges, particularly in dealing with hallucinations. In the context of establishing an LLM-based education system, this issue becomes more critical for providing seemingly correct yet incorrect answers, leading to misleading students. Drawing insights from approaches that introduce additional information from sources such as the web or textbooks can offer valuable lessons for the development of an LLM-based education system.

3.5.2 Domain-specific QA. Although LLMs are trained on vast corpora, they may still exhibit gaps in understanding specific domains. The primary challenge faced by LLMs in this task is the lack of domain knowledge. For specific domain questions, providing good answers often requires a considerable amount of expertise or skills in that field, while professional data is relatively scarce in the corpus of large-scale models. One straightforward approach is to fine-tune LLMs on specialized datasets. Typically, there are dedicated knowledge repositories for professional content that consolidate domain-specific knowledge, such as MedlinePlus[1], GeeksforGeeks[2], etc. Choi et al. [19] utilized an external knowledge base to generate a set of question-answer pairs and then employed fine-tuning to transfer financial knowledge to LLMs, significantly improving financial question-answering tasks. Another common approach is to leverage the in-context learning capability of LLMs by incorporating retrieved knowledge from the knowledge base into prompts. Peng et al. [100] demonstrate this approach in their work on pest identification. They first use text embeddings, which are dense vector representations of text, to retrieve relevant information from a knowledge base. Text embeddings allow for efficient and accurate retrieval of similar or related content based on the semantic similarity between the query and the stored information. Once the relevant knowledge is retrieved, it is incorporated into the prompts provided to the LLMs. The LLMs then utilize their automatic feature extraction capabilities to process and understand the retrieved information in the context of the pest identification task. zhang et al. [159] utilized K-nearest neighbors (KNN) [42] to

[1]https://medlineplus.gov
[2]https://www.geeksforgeeks.org

search for the most similar K records from an accounting database, serving as k-shot examples, and greatly improved accounting efficiency. There are also works that train and improve the retriever encoder [157], as well as distill and refine the data in the database [49]. Such retrieval frameworks have lower costs and can be more flexible in applications across different domains. Liu et al. [72] proposed the RETA-LLM, a system that leveraged an information retrieval system based on Google Search to initially retrieve the top-k documents relevant to a user's query, allowing LLMs to generate answers based on these retrieved documents. Furthermore, the system included plug-and-play modules that enable users to construct their own domain-specific LLMs. These modules covered various functionalities, including request rewriting, document retrieval, passage extraction, answer generation, and fact-checking.

By integrating Information Retrieval (IR) systems, LLMs can enhance their capabilities with professional knowledge, gaining valuable and precise supplemental information. Furthermore, according to Ren et al. [110], retrieval augmentation can also be employed to improve LLMs' ability to perceive facts within the boundaries of their legal knowledge, mitigating the issue of hallucinations. During the education process, different majors or courses involve different professional content. Applying external knowledge repositories as an enhancement mechanism can provide more accurate guidance in domain-specific contexts and mitigate the issues caused by misleading information. Therefore, domain-specific question-answering ability is crucial for developing an LLM-based education system.

3.5.3 Summary. While LLMs have mastered a broad range of open-world knowledge through extensive corpus training, their fixed parameters make it challenging to handle real-time, high-demand open-domain questions. Severe hallucination issues further compromise their accuracy in both open-domain and domain-specific queries. In the field of education, where authenticity is paramount, students may pose questions about textbook knowledge points. If the accuracy of these responses cannot be guaranteed, it could potentially mislead students. Therefore, a feasible solution to the inherent hallucination issues in foundational LLMs is to integrate external information, such as authoritative documents, allowing LLMs to base their responses on such external sources to mitigate hallucination problems.

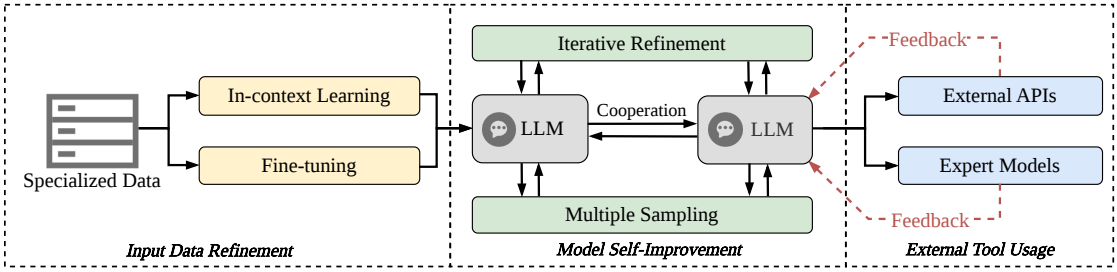


Fig. 3. A summary framework diagram for the approaches of LLMs in the development of education-related abilities. It categorizes previous enhancement strategies into three parts: Input Data Refinement, Model Self-Improvement, and External Tool Usage.

3.6 Discussion

Despite the varied contexts and specific challenges associated with each capability, certain strategies and insights resonate universally among researchers working to harness the capabilities of LLMs for educational purposes. Here we discuss the trends or commonalities we discovered across the development of capabilities.

3.6.1 High-quality data could help LLMs develop capabilities effectively. Since the advent of the deep learning era, high-quality training data has significantly improved model performance. This is especially true for LLMs, and the

approach extends beyond mere training. Through in-context learning, where desired inputs and outputs are formed into demonstrations and provided as prompts to LLMs, if the demonstrations are well-chosen, they can also greatly enhance the model's capabilities.

There is a notable trend towards leveraging smaller models fine-tuned on high-quality data to surpass the performance of larger language models. This approach emphasizes the importance of focused, domain-specific training rather than simply relying on the vast number of parameters in large-scale models. The term "high-quality data" here refers to data that provides detailed supervisory signals for specific domain problems. For instance, even powerful LLMs like GPT-4 encounter high error rates with large number multiplication in basic arithmetic problems. However, with a dataset that includes detailed steps for multiplication and addition, even a small transformer model can effectively solve these problems. Similarly, for reasoning tasks, it's possible to improve LLMs' performance on these issues without altering model parameters, simply by incorporating specific reasoning steps into the demonstrations in the prompt. This highlights that the application of "high-quality data" in LLMs extends beyond just fine-tuning.

There is a concerted effort among researchers to explore how smaller models can achieve equivalent or superior outcomes in certain scenarios where LLMs have already exceeded human benchmarks. This pursuit reflects a broader shift towards optimizing computational efficiency and model scalability, ensuring that the advancements in LLM technology remain accessible and sustainable. The primary scenarios and objectives for training small models can be divided into two categories:

- Training specialized models with high-quality data. For applications within specific, narrow fields, such as programming, developing a small, specialized model through data collection can facilitate deployment and reduce the computational resources needed for inference, among other benefits.
- Training small models through knowledge distillation. In cases where the required knowledge and skills for an application are more general, it may be challenging to construct a specific dataset for training a small model. By learning to match the outputs of the teacher model, the student model can effectively absorb the knowledge and skills of the larger model, without the need for a specific, curated dataset. This allows the smaller model to inherit the generalization capabilities of the large teacher model, while being more computationally efficient and easier to deploy in resource-constrained environments.

Efficiency is an unavoidable issue for LLMs during training and real-world deployment. Training-wise, low-loss, high-efficiency training schemes like LoRA (Low-Rank Adaptation) [46] are continually being introduced. These can significantly reduce the number of trainable parameters required during fine-tuning. However, small models are still needed. In practical applications, where the trained models need to be deployed and used for generating predictions or outputs, the size of the model still plays a critical role.

3.6.2 LLMs could achieve self-improvement. The inference abilities and text comprehension skills of LLMs enable them to conveniently obtain feedback, which allows them to refine their outputs. This process is known as the self-improvement, which is a general approach in improving LLMs' answers to all the capabilities. LLMs can achieve self-improvement through a methodical approach that involves iterative refinement and multiple sampling. For instance, to provide better responses to queries, an LLM may first generate an initial output. Then, it evaluates the output's effectiveness or accuracy. Leveraging multiple sampling, the LLM explores different solution pathways or creative responses, which expands its potential answers pool. Through iterative refinement, it compares, contrasts, and consolidates these possibilities, learning which strategies yield the best results. This could involve internal processes such as adjusting parameters based on feedback loops, where it might integrate data from new examples or corrections provided by human users.

Over time, this enhances the LLM's ability to provide more precise, informative, and contextually relevant answers, thus gradually improving its problem-solving and content creation skills. In addition to individual LLMs enhancing their capabilities through feedback, the method of coordinating multiple LLMs to improve overall output is also being explored across various fields. For example, in solving programming problems, the process can be segmented into different stages, with each stage managed by a distinct LLM. These LLMs communicate and collaborate to complete the task collectively. Alternatively, one LLM may act as a generator to produce answers, while another serves as an evaluator to provide feedback. Through continuous dialogue between the two, the response can be consistently assessed and refined, thereby improving the quality of the answer. This cooperative approach leverages the strengths of different models to achieve a more effective and sophisticated problem-solving mechanism. Although the method of self-improvement is effective, it often results in a longer time to produce responses. For an LLM-based education system, the importance of providing accurate answers to students' questions (to avoid misleading them) outweighs the need for speed. Therefore, employing a multi-agent approach to enhance answer quality through LLMs' collaboration, or using sampling and iterative optimization for self-improvement, is an appropriate strategy for developing an education system. This ensures that the system prioritizes the correctness and reliability of information, which is crucial in educational settings where students' learning effect is depend on the accuracy of the content provided.

3.6.3 Calling external tools is an universal method. The integration of external tools into the LLM framework is a widely adopted method. This strategy not only enhances the models' ability to access and incorporate real-time information and authoritative sources but also mitigates some of the inherent limitations of LLMs, such as their tendency towards factual inaccuracies or hallucinations. We can divide the use of external tools by LLMs into two perspectives:

- LLMs inherently have certain limitations that cannot be resolved through training alone, and external tools can be used to address these deficiencies. In this scenario, tools serve the LLMs. For example, LLMs' high error rates in large number multiplication can easily be mitigated by employing an external calculator. Similarly, LLMs' inability to access real-time information can be compensated for by retrieving the latest web pages via web API calls.
- Utilizing the reasoning and decision-making capabilities of LLMs, the invocation of external tools can influence the real world. In this approach, the primary task of LLMs is to make informed decisions about when and which tools to utilize in order to accomplish specific tasks. The key responsibility of LLMs in this approach is to act as intelligent agents that can analyze a given situation, understand the requirements and constraints, and determine the most appropriate course of action.

The LLM that solves problems by invoking external tools is a type of LLM agent, where the external tool is not necessarily an API but can also be an expert model. Fine-tuning LLMs on specific datasets can yield excellent results on the corresponding task, but it's impractical to fine-tune LLMs across datasets for all capabilities. A viable solution is to use fine-tuned small language models as expert models, which serve as external tools for the central LLM. Compared to APIs, the advantage of trained expert language models is their ability to understand more granular and flexible demands from the LLM, providing targeted feedback.

4 OVERALL DEVELOPMENT STATUS

Before exploring the possibilities of building an education system based on LLMs, we first need to investigate the performance of LLMs in capabilities related to education. We select representative benchmarks to assess the current development of LLMs across education-related capabilities. Specifically, we mainly collect the results from three sources:

Table 1. Overview of LLMs' performance on foundational education-related capabilities.

Models	Reference	Mathematics (GSM8K) (Pass@1)	Writing (OpenCompass) (Avg Score)	Programming (HumanEval) (Pass@1)	Reasoning (HelloSwag) (Acc)	Knowledge-based QA (TruthfulQA) (MC2)	General (C-Eval) (Avg Score)
GPT-4	OpenAI [97]	**92.00**	**62.00**	**67.00**	**91.40**	59.00	68.70
ChatGPT	OpenAI [96]	57.10	48.60	48.10	79.50	47.00	54.40
TigerBot-70B-Chat-V2(70B)	TigerResearch [125]	54.36	61.30	30.50	82.83	**75.40**	-
LLaMA2(70B)	Touvron et al. [127]	60.27	51.60	29.90	82.30	56.18	55.20
LLaMA(65B)	Touvron et al. [126]	43.37	47.10	23.70	82.30	55.09	38.80
Yi(34B)	lingyiwanwu [69]	50.64	48.90	26.20	82.00	56.23	**81.40**
Vicuna(33B)	TheVicunaTeam [123]	13.72	44.90	15.20	83.00	56.16	39.80
WizardLM(30B)	Xu et al. [142]	34.42	-	26.08	76.32	49.14	-
Moss(16B)	FudanUniversity [33]	6.90	39.00	-	55.80	49.00	33.10
Qwen(14B)	Bai et al. [7]	58.98	52.7	43.90	80.20	49.43	72.10
Baichuan2(13B)	Yang et al. [144]	55.30	51.50	17.07	66.90	48.98	40.00
Alpaca(7B)	Taori et al. [122]	0.15	39.50	9.10	75.71	36.28	29.90
ChatGLM3(6B)	Zeng et al. [153]	72.30	43.10	44.50	76.50	-	69.00

Huggingface[3], OpenCompass[4] and C-Eval[5]. The formal two are comprehensive leaderboards. C-Eval is a Chinese evaluation suite for foundation models spanning 52 diverse disciplines. We collect performance data from popular general LLMs on these benchmarks, and the compiled results are presented in Table 1, where one can observe that:

- It is hard for a single LLM to be superior across all capabilities. Among the current LLMs, GPT-4 has shown the most impressive overall performance. However, utilizing GPT-4 comes with higher associated costs compared to other LLMs, which can be a significant consideration for users and organizations with limited budgets, and it has been surpassed by TigerBot in knowledge-based QA tasks. For mathematics, GPT-4 achieves optimal performance on the representative dataset GSM8K, but it exhibits a higher error rate in basic arithmetic tasks, such as large number multiplication.
- LLMs still lag significantly behind humans in some crucial abilities. One notable example of this gap is illustrated by their performance on TruthfulQA [68], a benchmark designed to evaluate the ability of models to provide truthful and accurate answers, where human achieves achieving 94% accuracy while GPT-4 only got 59% correct.
- Most LLMs display considerable variation in developing these skills. While certain models (such as Alpaca and Yi) might excel in text comprehension tasks, their effectiveness often diminishes in areas requiring deep understanding and reasoning, like Mathematics and Programming. This reveals the substantial challenges in building a unified education-focused LLM since it may fail in certain areas.

5 POTENTIAL OF LLM-BASED EDUCATION SYSTEM

LLMs can potentially transform online education by understanding a wide range of student questions, similar to human teachers. They aim to provide support across different subjects and skill levels. With the latest developments in LLMs, we suggest two approaches for creating LLM-based education systems. The first involves training a comprehensive and unified LLM that can handle questions from various subjects. The second approach uses a mixture-of-experts (MoE) framework, integrating specialized models to support the system with an LLM controller to manage interactive dialogues with students.

[3]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard
[4]https://opencompass.org.cn/leaderboard-llm
[5]https://cevalbenchmark.com/static/leaderboard.html

5.1 Unified Approach

The most straightforward idea for establishing an LLM-based education system is to train a language model capable of answering students' questions across all subjects. As shown in Figure 4 (a), the foundational capabilities are included in the unified LLM, and the student can directly communicate with it and ask questions.

Research on whether general LLMs can handle educational tasks has been underway. Wang and Demszky [130] introduced three teacher coaching tasks for generative AI: (A) scoring transcript segments using classroom observation instruments, (B) identifying highlights and missed opportunities for effective instructional strategies, and (C) offering actionable suggestions to encourage more student reasoning. And evaluated by human teachers, ChatGPT on these tasks for elementary math classroom transcripts generates responses that are relevant to improving instruction, but they are often not novel or insightful. Beyond that, Phung et al. [102] assessed the programming education ability of ChatGPT and GPT-4 by comparing them with human tutors. The result shows that GPT-4 performs way better than ChatGPT, even close to human tutors in some scenarios, while it also highlights some situations in which GPT-4 struggles. In particular, for the grading feedback and task creation scenarios that have a substantial gap in the performance of GPT-4 compared to that of human tutors.

From the methods proposed by researchers in developing LLMs for educational capabilities, we can extract some common, scalable approaches to lay the groundwork for developing a unified LLM-based educational system:

- High-quality demonstrations. While collecting high-quality data from various fields for fine-tuning LLMs is impractical, achieving better responses through prompt engineering as a form of demonstration is feasible.
- API Tool Learning. For inherent challenges within LLMs, such as large number computations and the absence of real-time information, these can be addressed by incorporating external APIs as tools.
- Search-based methods. Attempts have been made across various fields to improve the task completion accuracy of LLMs using search-based methods, leveraging the probabilistic nature of LLMs. For challenging questions, LLMs might waver among multiple possible responses. Here, employing search-based methods to evaluate and filter all options can effectively enhance accuracy, offering a generally applicable solution.

The benefit of developing a unified LLM-based educational system is that it centers around a general LLM handling the core reasoning tasks. This setup means that all major language-based interactions are directly between the LLM and the students, making the system easier to deploy. The most significant effort and resources are invested during the training phase. This critical period is where the LLM gains expert-level skills in a wide range of subjects, preparing it to effectively support and educate students across various disciplines.

5.2 MoE Approach

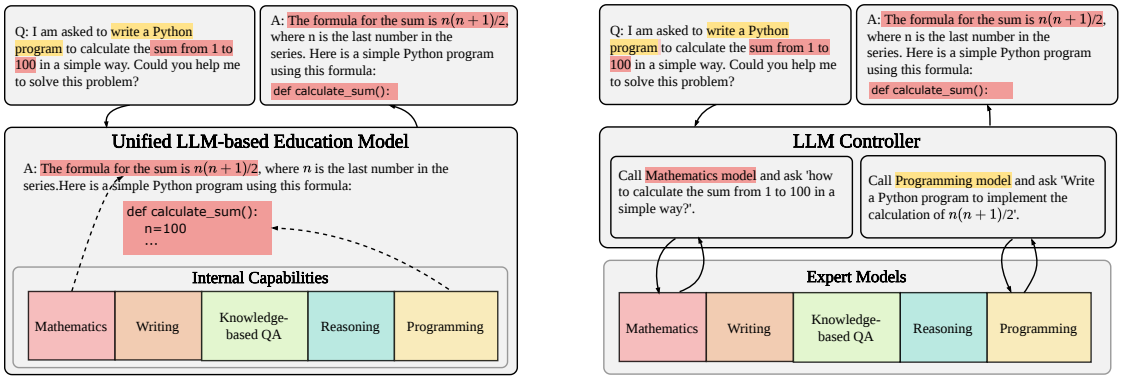
Section 3 reviewed the current development of LLMs across various capabilities. Unfortunately, despite the existence of comprehensive language models, such as GPT-4, these models often exhibit notable deficiencies in certain abilities. This situation poses a challenge, indicating that relying solely on an LLM itself for educational guidance involving all these capabilities is currently a difficult task. Yet, LLMs can achieve excellent results through fine-tuning individual capabilities, and their ability to comprehend human language is exceptionally strong. Therefore, we can aggregate models with distinct capabilities using a mixture-of-experts approach. By establishing an LLM-based controller for language interaction and task assignment with students, a currently feasible education system can be generated.

An education framework implemented with a mixture-of-experts (MoE) approach is illustrated in Figure 4(b), consisting of multiple models that excel in individual capabilities (not necessarily LLMs) and an LLM controller. The controller is mainly responsible for three tasks:

- Understand the student's request and decide which specific area or areas the request is about.
- Re-form the request to fit the input of the specific areas' expert models.
- Aggregate the output of the related experts and generate the final response to the student.

The advantage of the MoE approach is that training is less challenging. The result is a suite of models, each excelling in its specific domain or capability, which, when combined, offer a comprehensive educational tool. This specialization means training can be more focused and less onerous, optimizing resources towards developing excellence in distinct areas of knowledge and skills. However, one significant drawback is the increased potential for misunderstandings or errors during the system's inference phase, primarily due to the complexity of interactions between the different specialized models and the LLM controller. Errors can arise from the LLM controller misinterpreting student inputs or incorrectly assigning tasks to the specialized models. Moreover, integrating outputs from various experts into a coherent response can also introduce discrepancies, as differences in context or terminology used by each model can lead to inconsistencies in the overall communication with students.

Despite these challenges, the approach represents a practical pathway toward realizing an LLM-based educational assistant system. By leveraging specialized models for different capabilities, it's possible to create a more flexible and efficient system that can adapt to a wide range of educational needs and learning styles. The key to success lies in improving the integration and communication between the specialized models and the overarching LLM controller, ensuring that the system can handle complex inquiries and deliver accurate, useful responses to students. Currently, this approach appears to be a viable strategy for achieving the ambitious goal of an effective LLM-based educational assistant, promising a future where personalized education is accessible and adaptable to every learner's need.



(a) Unified Approach Sketch (b) MoE Approach Sketch

Fig. 4. Two frameworks towards LLM-based educational framework. (a) depicts the unified approach, where a single LLM addresses all aspects of educational-related queries, utilizing its internal capabilities such as mathematics, writing, knowledge-based question answering, reasoning, and programming. (b) illustrates the Mixture of Experts (MoE) approach, where an LLM controller is tasked with task distribution, delegating specific questions to specialized expert models that are proficient in individual areas.

6 CHALLENGES AND FUTURE DIRECTIONS

Recently, more and more researchers have been trying to apply LLMs to handle education tasks, such as course design, student evaluation, lesson plan design, and others. Nevertheless, there are still numerous challenges and opportunities that need to be addressed.

- **Planning for Students.** Solving subject-related questions for students can significantly address the issue of students not receiving targeted guidance from teachers. Furthermore, a higher-level task involves assessing students' knowledge status and planning their learning paths. These tasks are continuously evolving in the era of deep learning, with the adaptation and application of LLMs in these areas requiring further exploration. The primary challenge in planning learning paths for students lies in integrating knowledge from two aspects: first, the human knowledge system, which involves the structural relationships between knowledge points, requiring LLMs to understand the meaning of these knowledge points. Second, the personalized information of students, including their knowledge state, learning interests, and habits. Previous deep learning models for this task have been trained on sequences of student behaviors, which are often constructed as IDs rather than text. Since the foundation of LLMs is their ability to process text, the gap in data form presents a significant challenge in applying LLMs to this task.
- **Interdisciplinary Reasoning Ability.** Students may encounter interdisciplinary reasoning problems during real-world learning, requiring the education system to integrate multiple capabilities to formulate responses. As illustrated in Figure 1, the student intends to write a program to solve a mathematical problem, and the model needs first to comprehend the mathematical problem, devise a solution, and then generate the code. This process necessitates the model to synthesize both mathematical and programming capabilities. However, there is currently limited research in the integration of multiple interdisciplinary capabilities for LLMs at this stage, including both datasets and algorithms. Boyko et al. [11] examined how LLMs augment scientific inquiry, code development, scientific writing process, etc., and they propose that LLMs can foster interdisciplinary work by bridging knowledge gaps across scientific fields. However, they mainly discuss the LLMs' ability to help researchers' interdisciplinary collaboration instead of their ability to answer interdisciplinary questions. Cultivating an LLM to obtain this ability would help to develop a unified education system, which is an essential research direction.
- **Student Modeling.** Before the era of LLMs, in the age of deep learning, modeling student behavior was primarily achieved through sequential models, such as RNNs [117] and Transformers. A drawback of this approach was the inability to obtain student feedback, and the results lacked interpretability. Establishing an LLM-based education system allows students to articulate their personalized needs through dialog. Through such conversations, we can extract or infer personalized features about students, such as their current mastery of topics and preferences in learning styles. Besides modeling students from conversations, some researches [3, 6] have shown that LLMs have certain abilities in simulating humans and generating human samples. Applying to education, this ability indicates a potential for LLM-based student simulation. In this way, for the students with few interaction records, the LLM-based simulator could generate more samples and provide data to help the expert model better understand the student. It could help human teachers develop teaching skills better.
- **Social Bias of LLMs** Even after training through Reinforcement Learning from Human Feedback (RLHF) [98], LLMs can to some extent avoid answers that do not align with human cultural habits and values [30, 53], it has been observed that LLMs still manifest a certain degree of value bias in their responses. Feng et al. [30] pointed

out that the training of LLMs can lead to a certain degree of political bias. In the field of education, although most questions posed by students are related to scientific knowledge, issues such as writing and text reasoning should ideally be avoided by researchers developing foundational LLM models. In educational applications, the social biases inherent in LLMs pose a risk of inadvertently impart imparting skewed value systems to students. To safeguard the educational integrity and ensure the neutral and fair dissemination of knowledge, it is crucial to implement stringent measures. These could include developing advanced content review systems, establishing clear guidelines for the ethical use of LLMs in educational settings, and continuously monitoring the quality and nature of the LLM-generated content. Through these efforts, the educational community can leverage the benefits of LLMs while minimizing the risk of perpetuating biases, thus maintaining a balanced and objective learning environment.

- **Preventing Cheating in Education.** The texts generated by LLMs are indistinguishable from or even surpass those produced by humans in terms of fluency and usage. Although the primary aim of this article is to survey the development of LLMs in educational capacities, offering insights for the creation of an educational supermodel, it's crucial to recognize that in certain educational contexts, the over-reliance on LLMs is not desirable as it could hinder the natural learning process. For instance, while it's acceptable for students to seek assistance from LLMs to aid understanding during homework tasks, relying on LLMs to complete assignments without thoughtful consideration prevents students from receiving the necessary practice and learning. Therefore, identifying content generated by LLMs holds significant importance in the educational domain to prevent cheating and ensure the integrity of the learning process. By striking a balance between leveraging LLMs for educational enhancement and maintaining rigorous educational standards, educators and technologists can create an environment where students benefit from technology without compromising their learning journey. Recent studies have proposed detectors to identify LLM-generated texts. The foundational ideas could be broadly categorized into two primary strategies: statistical outlier detection methods and supervised classifiers. The former strategy focuses on uncovering statistical differences in linguistic features between texts written by humans and those generated by LLMs. This involves analyzing patterns, such as syntactic structures, vocabulary diversity, and stylistic nuances, that distinguish LLM-generated texts from human-crafted writings. These statistical indicators serve as markers for automated systems to detect content that deviates from human norms, potentially signaling LLM involvement. On the other hand, supervised classifiers rely on a different mechanism. This approach employs machine learning algorithms that have been trained on a labeled dataset containing examples of both human-written and LLM-generated texts. The battle against detecting LLM-generated texts is dynamic, necessitating ongoing research and adaptation of detection methodologies. As LLMs become increasingly sophisticated, the strategies for distinguishing their outputs from human-created content will need to evolve, embracing a combination of statistical insights, machine learning innovations, and perhaps new, yet-to-be-discovered methods.
- **Multi-modal Education.** In education, multi-modal information is common, like geometry problems combining text and images or textbook concepts with illustrations. Building a general intelligent education system requires handling such multi-modal data. Notably, the development of multi-modal LLMs is rapidly advancing [26, 149]. Different kinds of architectures and pre-train tasks are proposed [26]. However, the education domain often exhibits unique distribution characteristics in multi-modal information. Firstly, in education, images and text often have a high level of detail matching; for example, geometry questions often describe the specific parameters of shapes in images in great detail. Therefore, multi-modal large models need to have a high capability of capturing details in image information. Secondly, the multi-modal information in education often requires the

model to have a high capacity for cross-modal reasoning, but such data is less common in multi-modal datasets, leading to a potential shortfall in the reasoning capabilities of multi-modal language models across different modalities. Addressing this gap may require targeted datasets, and inspired by Chain of Thought (CoT) and its variants, the data should ideally contain detailed steps of multi-modal reasoning. Efforts are currently being made to address these data deficiencies in the field. Moreover, the characteristics of image and text data in education could limit the choice of structures for multi-modal models. For instance, a popular approach in the field of multi-modal large models involves dividing images into patches to create "image tokens", [88] which are then processed alongside text tokens as input. However, in the educational context, such division might disrupt certain key geometric structures within the images, thereby affecting their interpretation. This drawback could be more pronounced in educational multi-modal scenarios.

7 CONCLUSION

In this paper, we presented an overview of the development of the LLM-based education system. We first reviewed the important development of LLMs' education-related abilities. Then, we analyzed the potential of it and proposed two different ways of building such a system. We also highlighted the future directions that are worth working on. We hope this survey provides some insight into future research in this direction.

ACKNOWLEDGMENTS

The SJTU team is partially supported by National Natural Science Foundation of China (62177033).

REFERENCES

[1] Ghodai Abdelrahman, Qing Wang, and Bernardo Nunes. 2023. Knowledge tracing: A survey. *Comput. Surveys* 55, 11 (2023), 1–37.
[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).
[3] Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. 2023. Using large language models to simulate multiple humans and replicate human subject studies. In *International Conference on Machine Learning*. PMLR, 337–371.
[4] Mohammad AL-Smadi. 2023. ChatGPT and Beyond: The Generative AI Revolution in Education. *arXiv preprint arXiv:2311.15198* (2023).
[5] Md Adnan Arefeen, Biplob Debnath, and Srimat Chakradhar. 2023. Leancontext: Cost-efficient domain-specific question answering using llms. *arXiv preprint arXiv:2309.00841* (2023).
[6] Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. 2023. Out of one, many: Using language models to simulate human samples. *Political Analysis* 31, 3 (2023), 337–351.
[7] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609* (2023).
[8] Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Jean-Christophe Filliatre, Eduardo Gimenez, Hugo Herbelin, Gerard Huet, Cesar Munoz, Chetan Murthy, et al. 1997. *The Coq proof assistant reference manual: Version 6.1.* Ph. D. Dissertation. Inria.
[9] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687* (2023).
[10] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*. PMLR, 2206–2240.
[11] James Boyko, Joseph Cohen, Nathan Fox, Maria Han Veiga, Jennifer I Li, Jing Liu, Bernardo Modenesi, Andreas H Rauch, Kenneth N Reid, Soumi Tribedi, et al. 2023. An Interdisciplinary Outlook on Large Language Models for Scientific Research. *arXiv preprint arXiv:2311.04929* (2023).
[12] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *arXiv preprint arXiv:2307.03109* (2023).
[13] Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2022. CodeT: Code Generation with Generated Tests. arXiv:2207.10397 [cs.CL]

[14] Jiaqi Chen, Tong Li, Jinghui Qin, Pan Lu, Liang Lin, Chongyu Chen, and Xiaodan Liang. 2022. UniGeo: Unifying Geometry Logical Reasoning via Reformulating Mathematical Expression. *arXiv preprint arXiv:2212.02746* (2022).

[15] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[16] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. arXiv:2107.03374 [cs.LG]

[17] Sijia Chen, Baochun Li, and Di Niu. 2024. Boosting of thoughts: Trial-and-error problem solving with large language models. *arXiv preprint arXiv:2402.11140* (2024).

[18] Xianyu Chen, Jian Shen, Wei Xia, Jiarui Jin, Yakun Song, Weinan Zhang, Weiwen Liu, Menghui Zhu, Ruiming Tang, Kai Dong, et al. 2023. Set-to-sequence ranking-based concept-aware learning path recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5027–5035.

[19] Stephen Choi, William Gazeley, Siu Ho Wong, and Tingting Li. 2023. Conversational Financial Information Retrieval Model (ConFIRM). arXiv:2310.13001 [cs.IR]

[20] Albert T Corbett and John R Anderson. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction* 4 (1994), 253–278.

[21] Garett Cunningham, Razvan C Bunescu, and David Juedes. 2023. Towards Autoformalization of Mathematics and Code Correctness: Experiments with Elementary Proofs. *arXiv preprint arXiv:2301.02195* (2023).

[22] Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. 2015. The Lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*. Springer, 378–388.

[23] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. 2023. Active prompting with chain-of-thought for large language models. *arXiv preprint arXiv:2302.12246* (2023).

[24] Yangruibo Ding, Zijian Wang, Wasi Uddin Ahmad, Hantian Ding, Ming Tan, Nihal Jain, Murali Krishna Ramanathan, Ramesh Nallapati, Parminder Bhatia, Dan Roth, and Bing Xiang. 2023. CrossCodeEval: A Diverse and Multilingual Benchmark for Cross-File Code Completion. arXiv:2310.11248 [cs.LG]

[25] Chenhe Dong, Yinghui Li, Haifan Gong, Miaoxin Chen, Junxin Li, Ying Shen, and Min Yang. 2022. A Survey of Natural Language Generation. *ACM Comput. Surv.* 55, 8, Article 173 (dec 2022), 38 pages. https://doi.org/10.1145/3554727

[26] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. 2022. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936* (2022).

[27] Yaxin Fan, Feng Jiang, Peifeng Li, and Haizhou Li. 2023. GrammarGPT: Exploring Open-Source LLMs for Native Chinese Grammatical Error Correction with Supervised Fine-Tuning. In *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 69–80.

[28] Tao Fang, Shu Yang, Kaixin Lan, Derek F Wong, Jinpeng Hu, Lidia S Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *arXiv preprint arXiv:2304.01746* (2023).

[29] Huawen Feng, Yan Fan, Xiong Liu, Ting-En Lin, Zekun Yao, Yuchuan Wu, Fei Huang, Yongbin Li, and Qianli Ma. 2023. Improving Factual Consistency of Text Summarization by Adversarially Decoupling Comprehension and Embellishment Abilities of LLMs. *arXiv preprint arXiv:2310.19347* (2023).

[30] Shangbin Feng, Chan Young Park, Yuhan Liu, and Yulia Tsvetkov. 2023. From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair NLP models. *arXiv preprint arXiv:2305.08283* (2023).

[31] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. 2023. InCoder: A Generative Model for Code Infilling and Synthesis. arXiv:2204.05999 [cs.SE]

[32] Lingyue Fu, Huacan Chai, Shuang Luo, Kounianhua Du, Weiming Zhang, Longteng Fan, Jiayi Lei, Renting Rui, Jianghao Lin, Yuchen Fang, Yifan Liu, Jingkuan Wang, Siyuan Qi, Kangning Zhang, Weinan Zhang, and Yong Yu. 2023. CodeApex: A Bilingual Programming Evaluation Benchmark for Large Language Models. arXiv:2309.01940 [cs.CL]

[33] FudanUniversity. 2023. moss. https://github.com/OpenLMLab/MOSS.

[34] Wensheng Gan, Zhenlian Qi, Jiayang Wu, and Jerry Chun-Wei Lin. 2023. Large language models in education: Vision and opportunities. *arXiv preprint arXiv:2311.13160* (2023).

[35] Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, et al. 2023. G-llava: Solving geometric problem with multi-modal large language model. *arXiv preprint arXiv:2312.11370* (2023).

[36] Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487* (2020).

[37] Aritra Ghosh and Andrew Lan. 2021. Bobcat: Bilevel optimization-based computerized adaptive testing. *arXiv preprint arXiv:2108.07386* (2021).

[38] Jibing Gong, Shen Wang, Jinlong Wang, Wenzheng Feng, Hao Peng, Jie Tang, and Philip S Yu. 2020. Attentional graph convolutional networks for knowledge concept recommendation in moocs in a heterogeneous view. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 79–88.

[39] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yujiu Yang, Minlie Huang, Nan Duan, Weizhu Chen, et al. 2023. Tora: A tool-integrated reasoning agent for mathematical problem solving. *arXiv preprint arXiv:2309.17452* (2023).

[40] Grammarly. 2023. Grammarly. https://www.grammarly.com/

[41] TAL Education Group. 2024. MathGPT. https://www.mathgpt.com/ 2024-03-30.

[42] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. KNN model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings*. Springer, 986–996.

[43] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.

[44] Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071* (2022).

[45] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352* (2023).

[46] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

[47] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).

[48] iFLYTEK. 2024. AutoSpark. https://xinghuo.xfyun.cn/ 2024-03-30.

[49] Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. InPars-v2: Large Language Models as Efficient Dataset Generators for Information Retrieval. arXiv:2301.01820 [cs.IR]

[50] Albert Q Jiang, Wenda Li, and Mateja Jamnik. 2023. Multilingual Mathematical Autoformalization. *arXiv preprint arXiv:2311.03755* (2023).

[51] Albert Qiaochu Jiang, Wenda Li, Szymon Tworkowski, Konrad Czechowski, Tomasz Odrzygóźdź, Piotr Miłoś, Yuhuai Wu, and Mateja Jamnik. 2022. Thor: Wielding hammers to integrate language models and automated theorem provers. *Advances in Neural Information Processing Systems* 35 (2022), 8360–8373.

[52] Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. 2022. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283* (2022).

[53] Hang Jiang, Doug Beeferman, Brandon Roy, and Deb Roy. 2022. CommunityLM: Probing partisan worldviews from language models. *arXiv preprint arXiv:2209.07065* (2022).

[54] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics* 9 (2021), 962–977.

[55] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. 2024. RealTime QA: What's the Answer Right Now? *Advances in Neural Information Processing Systems* 36 (2024).

[56] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. 2023. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and individual differences* 103 (2023), 102274.

[57] Majeed Kazemitabaar, Xinying Hou, Austin Henley, Barbara Jane Ericson, David Weintrop, and Tovi Grossman. 2024. How Novices Use LLM-based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research* (<conf-loc>, <city>Koli</city>, <country>Finland</country>, </conf-loc>) *(Koli Calling '23)*. Association for Computing Machinery, New York, NY, USA, Article 3, 12 pages. https://doi.org/10.1145/3631802.3631806

[58] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172* (2019).

[59] Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115* (2022).

[60] Nayoung Lee, Kartik Sreenivasan, Jason D Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2023. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381* (2023).

[61] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).

[62] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. 2022. Solving quantitative reasoning problems with language models, 2022. *URL https://arxiv. org/abs/2206.14858* (2022).

[63] Nian Li, Chen Gao, Yong Li, and Qingmin Liao. 2023. Large language model-empowered agents for simulating macroeconomic activities. *arXiv preprint arXiv:2310.10436* (2023).

[64] Qingyao Li, Wei Xia, Li'ang Yin, Jian Shen, Renting Rui, Weinan Zhang, Xianyu Chen, Ruiming Tang, and Yong Yu. 2023. Graph Enhanced Hierarchical Reinforcement Learning for Goal-oriented Learning Path Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1318–1327.

[65] 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 preprint arXiv:2211.09110* (2022).

[66] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[67] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. 2023. How Can Recommender Systems Benefit from Large Language Models: A Survey. *arXiv preprint arXiv:2306.05817* (2023).

[68] Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958* (2021).

[69] lingyiwanwu. 2023. Yi. https://www.lingyiwanwu.com/.

[70] Bingchang Liu, Chaoyu Chen, Cong Liao, Zi Gong, Huan Wang, Zhichao Lei, Ming Liang, Dajun Chen, Min Shen, Hailian Zhou, Hang Yu, and Jianguo Li. 2023. MFTCoder: Boosting Code LLMs with Multitask Fine-Tuning. arXiv:2311.02303 [cs.LG]

[71] Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, et al. 2023. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295* (2023).

[72] Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. 2023. RETA-LLM: A Retrieval-Augmented Large Language Model Toolkit. arXiv:2306.05212 [cs.IR]

[73] Junling Liu, Ziming Wang, Qichen Ye, Dading Chong, Peilin Zhou, and Yining Hua. 2023. Qilin-Med-VL: Towards Chinese Large Vision-Language Model for General Healthcare. *arXiv preprint arXiv:2310.17956* (2023).

[74] Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks. *arXiv preprint arXiv:2305.14201* (2023).

[75] Yixin Liu, Budhaditya Deb, Milagro Teruel, Aaron Halfaker, Dragomir Radev, and Ahmed H Awadallah. 2022. On improving summarization factual consistency from natural language feedback. *arXiv preprint arXiv:2212.09968* (2022).

[76] Yixin Liu, Alexander R Fabbri, Jiawen Chen, Yilun Zhao, Simeng Han, Shafiq Joty, Pengfei Liu, Dragomir Radev, Chien-Sheng Wu, and Arman Cohan. 2023. Benchmarking generation and evaluation capabilities of large language models for instruction controllable summarization. *arXiv preprint arXiv:2311.09184* (2023).

[77] Yixin Liu, Alexander R Fabbri, Pengfei Liu, Dragomir Radev, and Arman Cohan. 2023. On Learning to Summarize with Large Language Models as References. *arXiv preprint arXiv:2305.14239* (2023).

[78] Yixin Liu, Pengfei Liu, Dragomir Radev, and Graham Neubig. 2022. BRIO: Bringing order to abstractive summarization. *arXiv preprint arXiv:2203.16804* (2022).

[79] Vadim Liventsev, Anastasiia Grishina, Aki Härmä, and Leon Moonen. 2023. Fully Autonomous Programming with Large Language Models. In *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM. https://doi.org/10.1145/3583131.3590481

[80] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts. *arXiv preprint arXiv:2310.02255* (2023).

[81] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610* (2022).

[82] Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2022. A survey of deep learning for mathematical reasoning. *arXiv preprint arXiv:2212.10535* (2022).

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

[84] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. WizardCoder: Empowering Code Large Language Models with Evol-Instruct. arXiv:2306.08568 [cs.CL]

[85] Qianou Ma, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. 2023. HypoCompass: Large-Language-Model-based Tutor for Hypothesis Construction in Debugging for Novices. arXiv:2310.05292 [cs.HC]

[86] Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. PROM: A Phrase-level Copying Mechanism with Pre-training for Abstractive Summarization. *arXiv preprint arXiv:2305.06647* (2023).

[87] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410* (2022).

[88] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. 2024. MM1: Methods, Analysis & Insights from Multimodal LLM Pre-training. *arXiv preprint arXiv:2403.09611* (2024).

[89] Rob R Meijer and Michael L Nering. 1999. Computerized adaptive testing: Overview and introduction. *Applied psychological measurement* 23, 3 (1999), 187–194.

[90] Jesse G Meyer, Ryan J Urbanowicz, Patrick CN Martin, Karen O'Connor, Ruowang Li, Pei-Chen Peng, Tiffani J Bright, Nicholas Tatonetti, Kyoung Jae Won, Graciela Gonzalez-Hernandez, et al. 2023. ChatGPT and large language models in academia: opportunities and challenges. *BioData Mining*

16, 1 (2023), 20.

[91] Osamah Mohammed, Thaeer Mueen Sahib, Israa M Hayder, Sani Salisu, Misbah Shahid, et al. 2023. ChatGPT Evaluation: Can It Replace Grammarly and Quillbot Tools? *British Journal of Applied Linguistics* 3, 2 (2023), 34–46.

[92] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. arXiv:2203.13474 [cs.LG]

[93] Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. 2002. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer.

[94] Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114* (2021).

[95] Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR–grammatical error correction: tag, not rewrite. *arXiv preprint arXiv:2005.12592* (2020).

[96] OpenAI. 2023. chatgpt. https://chat.openai.com/.

[97] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

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

[99] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? *arXiv preprint arXiv:2103.07191* (2021).

[100] Ruoling Peng, Kang Liu, Po Yang, Zhipeng Yuan, and Shunbao Li. 2023. Embedding-based Retrieval with LLM for Effective Agriculture Information Extracting from Unstructured Data. arXiv:2308.03107 [cs.AI]

[101] Shuai Peng, Di Fu, Yijun Liang, Liangcai Gao, and Zhi Tang. 2023. GeoDRL: A Self-Learning Framework for Geometry Problem Solving using Reinforcement Learning in Deductive Reasoning. In *Findings of the Association for Computational Linguistics: ACL 2023*. 13468–13480.

[102] Tung Phung, Victor-Alexandru Pădurean, José Cambronero, Sumit Gulwani, Tobias Kohn, Rupak Majumdar, Adish Singla, and Gustavo Soares. 2023. Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors. arXiv:2306.17156 [cs.CY]

[103] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J Guibas, and Jascha Sohl-Dickstein. 2015. Deep knowledge tracing. *Advances in neural information processing systems* 28 (2015).

[104] Stanislas Polu and Ilya Sutskever. 2020. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393* (2020).

[105] Xiao Pu, Mingqi Gao, and Xiaojun Wan. 2023. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558* (2023).

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

[107] 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. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

[108] Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. CoEdIT: Text Editing by Task-Specific Instruction Tuning. *arXiv preprint arXiv:2305.09857* (2023).

[109] Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361* (2019).

[110] Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the Factual Knowledge Boundary of Large Language Models with Retrieval Augmentation. arXiv:2307.11019 [cs.CL]

[111] Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950* (2023).

[112] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs.CL]

[113] Laura Ruis, Akbir Khan, Stella Biderman, Sara Hooker, Tim Rocktäschel, and Edward Grefenstette. 2022. Large language models are not zero-shot communicators. *arXiv preprint arXiv:2210.14986* (2022).

[114] Tomohiro Sawada, Daniel Paleka, Alexander Havrilla, Pranav Tadepalli, Paula Vidas, Alexander Kranias, John J Nay, Kshitij Gupta, and Aran Komatsuzaki. 2023. Arb: Advanced reasoning benchmark for large language models. *arXiv preprint arXiv:2307.13692* (2023).

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

[116] Chenhui Shen, Liying Cheng, Xuan-Phi Nguyen, Yang You, and Lidong Bing. 2023. Large language models are not yet human-level evaluators for abstractive summarization. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 4215–4233.

[117] Alex Sherstinsky. 2020. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena* 404 (2020), 132306.

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

[119] Dominik Sobania, Martin Briesch, Carol Hanna, and Justyna Petke. 2023. An analysis of the automatic bug fixing performance of chatgpt. In *2023 IEEE/ACM International Workshop on Automated Program Repair (APR)*. IEEE, 23–30.

[120] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems* 33 (2020), 3008–3021.

[121] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937* (2018).

[122] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. alpaca. https://crfm.stanford.edu/2023/03/13/alpaca.html.

[123] TheVicunaTeam. 2023. vicuna. https://lmsys.org/blog/2023-03-30-vicuna/.

[124] Nathan A Thompson and David A Weiss. 2019. A framework for the development of computerized adaptive tests. *Practical Assessment, Research, and Evaluation* 16, 1 (2019), 1.

[125] TigerResearch. 2023. Tigerbot. https://github.com/TigerResearch/TigerBot.

[126] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[127] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[128] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2022. Large Language Models Still Can't Plan (A Benchmark for LLMs on Planning and Reasoning about Change). *arXiv preprint arXiv:2206.10498* (2022).

[129] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, et al. 2023. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214* (2023).

[130] Rose E Wang and Dorottya Demszky. 2023. Is ChatGPT a Good Teacher Coach? Measuring Zero-Shot Performance For Scoring and Providing Actionable Insights on Classroom Instruction. *arXiv preprint arXiv:2306.03090* (2023).

[131] Shen Wang, Tianlong Xu, Hang Li, Chaoli Zhang, Joleen Liang, Jiliang Tang, Philip S Yu, and Qingsong Wen. 2024. Large Language Models for Education: A Survey and Outlook. *arXiv preprint arXiv:2403.18105* (2024).

[132] Xiaoxuan Wang, Ziniu Hu, Pan Lu, Yanqiao Zhu, Jieyu Zhang, Satyen Subramaniam, Arjun R Loomba, Shichang Zhang, Yizhou Sun, and Wei Wang. 2023. Scibench: Evaluating college-level scientific problem-solving abilities of large language models. *arXiv preprint arXiv:2307.10635* (2023).

[133] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).

[134] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. CodeT5+: Open Code Large Language Models for Code Understanding and Generation. arXiv:2305.07922 [cs.CL]

[135] Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*. 845–854.

[136] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[137] Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *arXiv preprint arXiv:2303.13648* (2023).

[138] Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Advances in Neural Information Processing Systems* 35 (2022), 32353–32368.

[139] Chunqiu Steven Xia and Lingming Zhang. 2023. Conversational automated program repair. *arXiv preprint arXiv:2301.13246* (2023).

[140] Huajian Xin, Haiming Wang, Chuanyang Zheng, Lin Li, Zhengying Liu, Qingxing Cao, Yinya Huang, Jing Xiong, Han Shi, Enze Xie, et al. 2023. LEGO-Prover: Neural Theorem Proving with Growing Libraries. *arXiv preprint arXiv:2310.00656* (2023).

[141] Jing Xiong, Jianhao Shen, Ye Yuan, Haiming Wang, Yichun Yin, Zhengying Liu, Lin Li, Zhijiang Guo, Qingxing Cao, Yinya Huang, et al. 2023. TRIGO: Benchmarking Formal Mathematical Proof Reduction for Generative Language Models. *arXiv preprint arXiv:2310.10180* (2023).

[142] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. WizardLM: Empowering Large Language Models to Follow Complex Instructions. arXiv:2304.12244 [cs.CL]

[143] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is inevitable: An innate limitation of large language models. *arXiv preprint arXiv:2401.11817* (2024).

[144] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, et al. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305* (2023).

[145] Kaiyu Yang, Aidan M Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. 2023. Leandojo: Theorem proving with retrieval-augmented language models. *arXiv preprint arXiv:2306.15626* (2023).

[146] Zhen Yang, Ming Ding, Qingsong Lv, Zhihuan Jiang, Zehai He, Yuyi Guo, Jinfeng Bai, and Jie Tang. 2023. GPT Can Solve Mathematical Problems Without a Calculator. *arXiv preprint arXiv:2309.03241* (2023).

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

[148] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378* (2021).

[149] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Haowei Liu, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. 2023. mPLUG-Owl2: Revolutionizing Multi-modal Large Language Model with Modality Collaboration. *arXiv preprint arXiv:2311.04257* (2023).

[150] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284* (2023).

[151] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. 2023. How well do Large Language Models perform in Arithmetic tasks? *arXiv preprint arXiv:2304.02015* (2023).

[152] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. 2022. Star: Bootstrapping reasoning with reasoning, 2022. *URL https://arxiv.org/abs/2203.14465* (2022).

[153] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414* (2022).

[154] Beichen Zhang, Kun Zhou, Xilin Wei, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and Improving Tool-Augmented Computation-Intensive Math Reasoning. *arXiv preprint arXiv:2306.02408* (2023).

[155] Jiaxin Zhang, Zhongzhi Li, Mingliang Zhang, Fei Yin, Chenglin Liu, and Yashar Moshfeghi. 2024. GeoEval: Benchmark for Evaluating LLMs and Multi-Modal Models on Geometry Problem-Solving. *arXiv preprint arXiv:2402.10104* (2024).

[156] Ming-Liang Zhang, Fei Yin, and Cheng-Lin Liu. 2023. A Multi-Modal Neural Geometric Solver with Textual Clauses Parsed from Diagram. *arXiv preprint arXiv:2302.11097* (2023).

[157] Peitian Zhang, Shitao Xiao, Zheng Liu, Zhicheng Dou, and Jian-Yun Nie. 2023. Retrieve Anything To Augment Large Language Models. arXiv:2310.07554 [cs.IR]

[158] Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510* (2023).

[159] Sabrina zhang, Daksha Yadav, and Tom Jin. 2023. Cash transaction booking via retrieval augmented LLM. In *KDD 2023 Workshop on Robust NLP for Finance (RobustFin)*. https://www.amazon.science/publications/cash-transaction-booking-via-retrieval-augmented-llm

[160] Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, et al. 2023. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification. *arXiv preprint arXiv:2308.07921* (2023).

[161] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2023. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406* (2023).

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

[163] Shuyan Zhou, Uri Alon, Frank F Xu, Zhiruo Wang, Zhengbao Jiang, and Graham Neubig. 2022. Docprompting: Generating code by retrieving the docs. *arXiv preprint arXiv:2207.05987* (2022).