https://doi.org/10.1007/s11704-025-50058-z

REVIEW ARTICLE

A comprehensive taxonomy of prompt engineering techniques for large language models

Yao-Yang LIU[1], Zhen ZHENG[2], Feng ZHANG[1]✉, Jin-Cheng FENG[1], Yi-Yang FU[1], Ji-Dong ZHAI[3], Bing-Sheng HE[4], Xiao ZHANG[1], Xiao-Yong DU[1]

1. Key Laboratory of Data Engineering and Knowledge Engineering (MOE), School of Information, Renmin University of China, Beijing 100872, China
2. Microsoft AI, Microsoft Asia Pacific Research and Development Group, Beijing 100080, China
3. Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
4. School of Computing, National University of Singapore, Singapore 119077, Singapore

Received January 21, 2025; accepted March 19, 2025
E-mail: fengzhang@ruc.edu.cn

Abstract
Large Language Models (LLMs) have demonstrated remarkable performance across various downstream tasks, as evidenced by numerous studies. Since 2022, generative AI has shown significant potential in diverse application domains, including gaming, film and television, media, and finance. By 2023, the global AI-generated content (AIGC) industry had attracted over fanxiexian_myfh26 billion in investment. As LLMs become increasingly prevalent, prompt engineering has emerged as a key research area to enhance user-AI interactions and improve LLM performance. The prompt, which serves as the input instruction for the LLM, is closely linked to the model's responses. Prompt engineering refines the content and structure of prompts, thereby enhancing the performance of LLMs without changing the underlying model parameters. Despite significant advancements in prompt engineering, a comprehensive and systematic summary of existing techniques and their practical applications remains absent. To fill this gap, we investigate existing techniques and applications of prompt engineering. We conduct a thorough review and propose a novel taxonomy that provides a foundational framework for prompt construction. This taxonomy categorizes prompt engineering into four distinct aspects: profile and instruction, knowledge, reasoning and planning, and reliability. By providing a structured framework for understanding its various dimensions, we aim to facilitate the systematic design of prompts. Furthermore, we summarize existing prompt engineering techniques and explore the applications of LLMs across various domains, highlighting their interrelation with prompt engineering strategies. This survey underscores the progress of prompt engineering and its critical role in advancing AI applications, ultimately aiming to provide a systematic reference for future research and applications.

Keywords
prompt engineering; large language models; AI agents; survey; taxonomy

1 Introduction

Large Language Models (LLMs), such as GPT-4, have attracted considerable attention due to their advanced capabilities in language comprehension and generation [1–12]. Users can effectively leverage the diverse competencies of LLMs by employing task-specific instructions, or prompts [13–16]. However, despite their impressive abilities, LLMs encounter several challenges in practical applications. For instance, while Reinforcement Learning from Human Feedback (RLHF) training enhances LLM conversational skills, making them more human-like, this anthropomorphic assumption does not universally apply to all LLMs [17]. As a result, informal prompts often yield responses from LLMs that lack the professionalism and precision expected in various contexts. The generalization capabilities of LLMs can lead to highly variable outputs when presented with unstructured prompts. Additionally, LLMs are susceptible to the "hallucination" problem, where generated content may include logical fallacies, fabricated facts, and data-driven biases. Consequently, designing prompts that yield more professional, stable, and reliable responses remains a significant challenge.

To address these challenges, extensive research has been dedicated to developing and refining prompt engineering techniques, which are essential for enhancing the performance of LLMs across various tasks and application domains. Researchers have explored diverse methodologies, including role-play prompting [18–20], task instruction [21], knowledge augmentation [22–26], and recursive prompting for reasoning [27–29]. For instance, Wei et al. [30] introduced a linear method for recursive prompting known as the "Chain-of-Thought" (CoT) approach. Although the CoT prompting technique has shown promising results, its inherent greedy strategy and linear structure introduce significant limitations, especially in scenarios that demand nuanced reasoning and greater flexibility. In response, Long [31] developed the Tree-of-Thought approach, which utilizes a tree-based structure. Similarly, Yao et al. [32] proposed the Graph-of-Thought approach, leveraging graph structures to enhance reasoning capabilities. As the application of system-level techniques in prompt engineering expands, LangChain[1)], a comprehensive framework integrates prompt templates with various system components to optimize prompts for LLMs. Although considerable research has been conducted on prompt engineering, a comprehensive review of the topic remains lacking.

1.1 Related surveys

The closest to our work are surveys on prompt learning and prompt engineering. Liu et al. [33] proposed the concept of prompt-based learning and provided an overview of fundamental principles, presenting a comprehensive range of prompt forms, including soft and hard prompts. Wei et al. [34] explored LLM performance under different prompt techniques, emphasizing its emergent abilities while reviewing challenges and outlining future directions for enhancing model performance through prompt engineering. With the advancement of LLMs, researchers have increasingly focused on hard prompt-based engineering techniques for these models. Recently, Sahoo et al. [35] has provided a description of the techniques and applications of prompt engineering, dividing prompt engineering techniques and applications into 12 categories within fixed scenarios. Li et al. [36] summarized goal-oriented prompt engineering techniques based on human reasoning principles. Some surveys concentrate on specific aspects, such as the study [28], which explores techniques aimed at enhancing reasoning performance, while Yu et al. [29] discussed these techniques from a philosophical reasoning perspective. Meanwhile, Mialon et al. [37] centered on prompt augmentation.

Synergy. While existing surveys focus on summarizing prompt engineering techniques within fixed scenarios, much less has been done to provide a general framework for prompt engineering. Prompt engineering is applied in numerous scenarios. Hence, classification fixed to a specific scenario complicates its migration to new application tasks. This calls for a taxonomy that encompasses the entire pipeline for LLM applications across various use cases. In this paper, we propose a universal prompt engineering taxonomy that outlines the pipeline for designing effective prompts. This taxonomy categorizes prompt engineering techniques from the perspective of their underlying principles. Building on this taxonomy, we correlate various applications of LLMs with prompt engineering techniques. Furthermore, this taxonomy encompasses both fundamental and advanced prompt techniques, offering detailed guidance on prompt design. With the continuous advancement of prompt engineering techniques, this taxonomy can be further extended.

1.2 Our contribution

We restrict our focus to discrete prefix prompts [38] rather than cloze prompts [39,40], as modern LLM architectures (specifically decoder-only models) demonstrate superior performance in various fields. Furthermore, our study concentrates on hard (discrete) prompts [33,41] rather than soft (continuous) prompts [33,42]. Finally, we refine the scope of prompt engineering as the technology to modify prompt content, allowing us to cover the full range of user and system-level technologies. To the best of our knowledge, no existing survey reviews prompt engineering techniques from a principled perspective and summarizes their applications. In contrast to previous surveys, the primary contributions of this paper are as follows:

- This survey represents the first comprehensive analysis of prompt engineering, approached from the perspective of human problem-solving principles. It includes an in-depth exploration of the field's background, taxonomy, applications, and a summary of prompt engineering.
- We present a comprehensive taxonomy of prompt engineering across four aspects: profile and instruction, knowledge, reasoning and planning, and reliability. This taxonomy provides a foundational framework encompassing essential building blocks and methodological abstractions crucial for prompt engineering.
- We summarize existing typical and state-of-the-art studies according to their domains, providing a convenient reference for researchers and developers.
- We generalize the taxonomy of methodologies as design factors for successful prompt engineering and propose a reasonable process flow for designing prompts.

2 Background

Prompt engineering is the technique of guiding model output through the strategic design of task-specific instructions (prompts) without altering model parameters. This approach has gained traction as a means to enhance the performance of LLMs across various tasks and fields. Research [43,44] indicates that the text input to LLMs significantly influences their effectiveness in downstream tasks. Therefore, prompt engineering serves as a strategic tool to guide model outputs. Given the intrinsic link between prompts and LLM functions, it is essential to design prompts with a nuanced understanding of LLM capabilities.

2.1 LLMs' Basic capabilities

Understanding the diverse capabilities of LLMs is crucial for creating effective and task-specific prompts. Existing literature [45] highlights that LLMs exhibit a broad range of capabilities, including reasoning, instruction-following, and in-context learning (ICL).

Reasoning is a core ability of LLMs, enabling them to perform complex multi-step reasoning through specific prompt design methods. This fundamental problem-solving ability supports various practical applications, such as medical diagnosis, legal decisions, and virtual assistants. Instruction-following refers to the LLM completing

[1)] LangChain-ai. See www.langchain.com/ website, 2024.

a new task according to task instructions without using an example. The generalization ability of LLMs allows them to generate, summarize, extract, classify, and rewrite text based on designed instructions.

In-context learning (ICL) is a paradigm that allows LLMs to perform tasks by learning from a few examples provided within the context [46]. This approach diverges from conventional supervised learning, which requires extensive training datasets and model parameter updates. The essence of ICL lies in its ability to leverage analogy and pattern recognition. When faced with a new query, LLMs refer to demonstration examples in the context to identify and apply relevant patterns without altering their underlying parameters. This process is akin to how humans learn from a few instances and generalize knowledge to new situations. The model's predictions are made by concatenating the query with contextual examples and using a scoring function to determine the most likely outcome. Prompt engineering primarily leverages ICL capabilities to steer the output of LLMs by carefully designing the content and structure of the input context.

2.2 Classification

To systematically categorize and generalize various prompt engineering techniques, we propose a novel taxonomy based on the functional divisions within agent systems. This approach aims to ensure both comprehensiveness and orthogonality in our taxonomy. Current research in agent systems divides agent functions into four main categories: profile function, memory function, planning function, and action function [47–49]. This comprehensive division ensures that any workflow based on LLMs can be covered by these four aspects.

Building on this functional division, we align prompt engineering workflows with these categories. We classify prompt engineering into four aspects:

- Profile and instruction. This foundational aspect defines the basic attributes and scenarios for LLMs. It standardizes LLM responses based on the information provided through natural language prompts. Profile and instruction examines prompt design from the perspective of textual content, proposing a foundational framework for basic prompt construction. Advanced prompt engineering techniques are developed based on this foundational framework.
- Knowledge. This aspect involves incorporating information from a local database into the prompt to mitigate the "hallucination" problem and improve the professionalism of the LLM. Although Knowledge techniques generally enhance LLMs in most cases, it is still necessary to assess whether knowledge augmentation is required based on the specific task.
- Reasoning and planning. This approach enhances the reasoning ability of LLMs. It includes decomposing goals, such as Chain of Thought [30], and utilizing tools and feedback to facilitate reasoning. This is crucial for improving LLM performance when handling complex tasks.
- Reliability. This aspect refers to the process of reducing bias in LLM responses. It involves ensuring both the stability of content generated multiple times (content bias) and the generation of content that is free of bias, stereotypes, or cultural impairment (value bias). Reliability is a crucial step in prompt design, and it can be combined with other prompt engineering techniques to enhance the performance of LLMs in real-world tasks.

Synergy. Our taxonomy provides a universal method for prompt design. Within this framework, profile and instruction represent the initial step in prompt design, establishing a foundational prompt suitable for daily use. Reliability serves as the final step, ensuring the stability and safety of the model's responses, which is essential for the practical application of prompt engineering. Both knowledge and reasoning and planning are techniques that enhance foundational prompts, improving the performance of LLMs through knowledge augment, target decomposition, and self-feedback. The application of these enhancement techniques depends on the specific use case. To date, our four categories encompass the vast majority of prompt engineering techniques. It is anticipated that future advancements in prompt engineering will focus on enhancing prompts. Due to the generality of our taxonomy, it can be expanded accordingly.

2.3 Application

In Subsection 2.2, we introduce a comprehensive taxonomy for prompt engineering, categorizing the process of prompt construction in practical applications into four distinct components. Prompt engineering facilitates the practical deployment of LLMs across various applications. In Section 4, we aim to provide a thorough classification of the application areas of LLMs. To align with our proposed taxonomy, we categorize the application fields of LLMs into two main areas: Cognitive Applications of LLMs and Transformative Applications of LLMs.

1) Cognitive applications of LLMs

Cognitive applications of LLMs refer to instances where users leverage LLMs to acquire or process information. The extensive number of parameters in LLMs endows them with vast knowledge and robust capabilities for knowledge processing. Based on the types of results generated by LLMs, cognitive applications can be classified into two categories.

- Information acquisition. This process involves extracting pertinent knowledge from LLMs through structured dialogues. The LLM outputs knowledge that meets task requirements. Common applications include chatbots and professional knowledge Q&A systems, where LLMs assist users in obtaining necessary information. In computer science, LLMs enhance system performance in search engines and training data augmentation.
- In-depth information analysis. This application leverages LLMs to analyze and reason over user input data, generating conclusions derived from the underlying information. In fields such as financial markets and chemical analysis, LLMs perform comprehensive analyses of complex data from multiple dimensions, deriving valuable insights and summaries.

2) Transformative applications of LLMs

Transformative Applications of LLMs refer to scenarios where LLMs autonomously execute task processes, reducing the need for human intervention. Based on the types of tasks executed by the LLM and the structure of prompt organization, transformative tasks can be categorized into two categories.

- Physical world applications. By employing prompt engineering techniques, such as Task Information (Subsection 3.1) and RAG (Subsection 3.2), LLMs can autonomously execute tasks in fields such as software engineering [50,51] and robotics [52]. With advancements in prompt engineering methodologies, LLMs have the potential to undertake specialized professional roles, such as AI-driven doctors [53] and AI legal advisors [54,55].
- Creative applications. By leveraging prompt engineering techniques, LLMs can generate literature and music [56–58], effectively replacing humans in the creative workflow. With the advent of increasingly multimodal LLMs, the application of these models in video creation has emerged as a significant area of development [59].

3 Taxonomy

This section provides an overview of current prompt engineering techniques in the context of generative model prompting. As illustrated in Fig. 1, we refine these techniques based on the distinctive features of different stages of the processing task.

3.1 Profile and instruction

Insight: Profile and Instruction summarizes prompt engineering techniques based on textual content. It defines the LLM's fundamental attributes and task specifications by designing prompts that incorporate personality information, task descriptions, and few-shot examples. As the first step in constructing prompts, it establishes a framework for more advanced prompt engineering approaches.

The primary purpose of this line of work is to determine a reasonable knowledge location for the LLM [60], concretely embodied in personality information (Seubsction 3.1), task information (Subsection 3.1), and demonstration information (Subsection 3.1), as shown in Fig. 2.

1) Personality information

LLMs can perform tasks by adopting specific roles, such as lawyers, teachers, and domain experts [18,19]. Personality information is a crucial part of the profile, defining the attributes of the LLM's role. These features are often included at the start of prompts to shape the LLM's response. Typically, personality information covers key characteristics like age, gender, and profession [20], along with tone and psychological traits, thereby reflecting the personality of the LLM's role. Assigning a specific role to an LLM establishes a suitable starting point for its responses. For instance, texts containing information about law are more likely to appear in the context of the word "lawyer". As a result, law-related information receives more attention along with the word "lawyer" during the inference stage, leading to responses pertinent to legal matters.

2) Task information

Task information plays a pivotal role in prompt design. Providing clear and well-defined task instructions enables LLMs to produce more specialized and contextually relevant outputs.

Task instruction. Task instruction is a prompt technique used to standardize the output of LLMs by incorporating clear task objectives and requirements into the prompt. This approach can be summarized into three key aspects: intent, domain, and demand. As shown in Fig. 3, intent defines the task's goal, demand specifies the detailed requirements, and domain identifies the information source relevant to the task. Research [61] has shown that when fundamental instructions lack clarity, LLM responses tend to be overly general. If

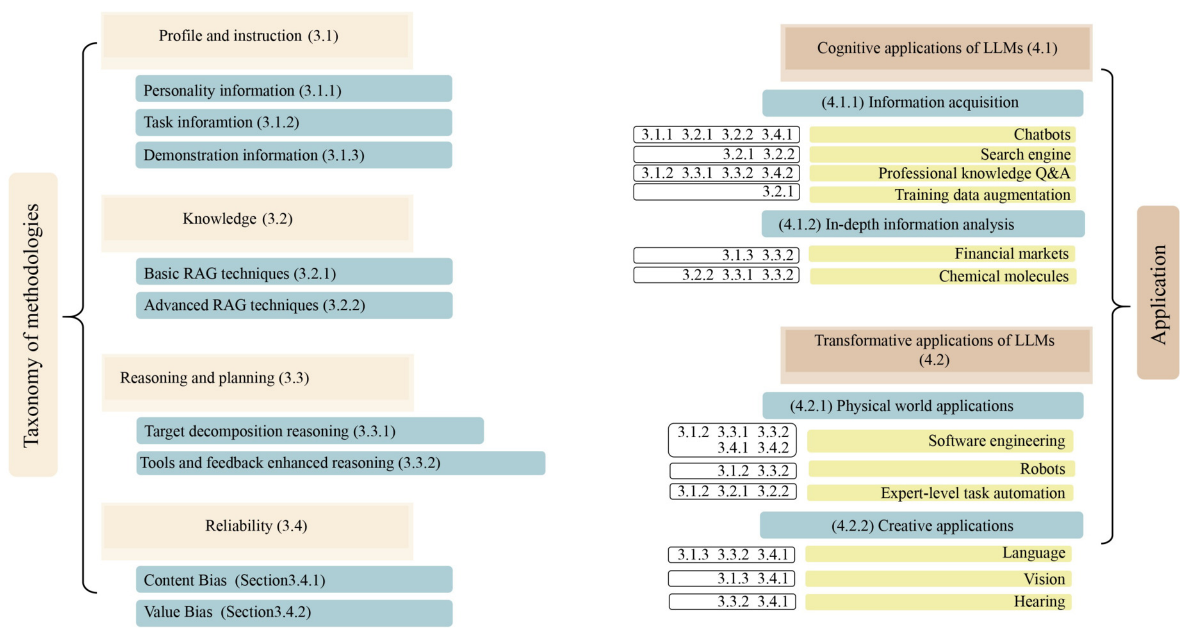

Fig. 1 The taxonomy tree of prompt engineering methodologies. The application tree on the right reveals the relationship between the practical application of LLMs and our categorized prompt engineering techniques (the numbers in parentheses stand for the corresponding subsections)

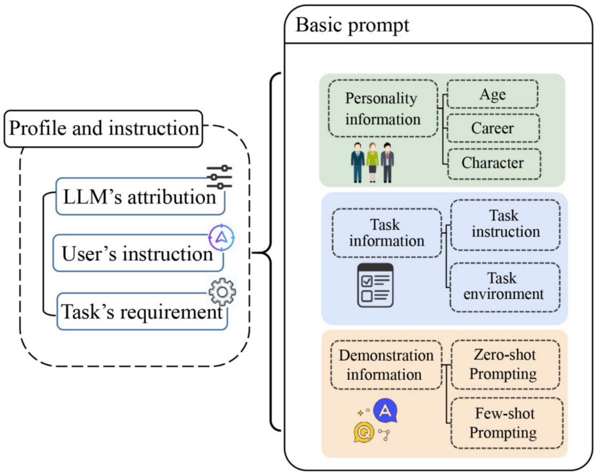

Fig. 2 Profile and instruction categorizes basic prompts into three components: personality information, task information, and demonstration information. This framework provides a foundational structure for prompt design in general-purpose applications

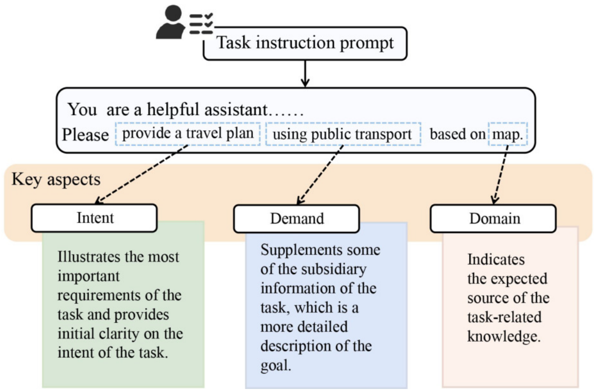

Fig. 3 An example prompt of the task instruction paradigm. The prompt consists of intent, demand, and domain, providing the basic information elements of the task

the prompt's structure is ambiguous and the content is too broad, the LLM faces numerous options, making it difficult to focus on the critical parts of the prompt. Consequently, this can lead to extensive but unfocused results. Studies such as [21] suggest that instruction understanding is a promising alternative paradigm for few-shot learning. Compared to examples, instructions provide stronger expressiveness and more stringent constraint capabilities [21].

Task environment. Environment Information incorporates task environment details into the prompt depending on the type of task. Providing information regarding the virtual task environment of the LLM can facilitate response generation, ensuring alignment with task requirements. For instance, if the LLM is tasked with "Fetch a bottle from the kitchen", the LLM+P model [43] can incorporate information about the task environment and the cost of the action, such as the distance from the current location to the kitchen and the location of the objects that the task needs to interact with in the prompt. This facilitates the generation of a more feasible plan, represented using a PDDL (Planning Domain Definition Language) framework [43]. However, environmental information alone is not sufficient. The LLM should also learn the anticipated consequences of its actions and verify if the current environment meets the conditions specified in the prompt. DEPS [62] addresses this by describing the objects accessible to the LLM and defining an action format that outlines both the conditions and potential outcomes of those actions. This approach enables the LLM to understand the relationships between elements in the task environment, comprehend the environmental information, and generate responses consistent with the task requirements. These methods have shown significant improvements over traditional deep learning techniques in open-world scenarios, such as Minecraft [63].

3) Demonstration information

Demonstration information is a technique that involves adding specific input-output mappings to the prompt. Research [64] indicates that LLMs trained on sufficiently large and diverse datasets provide responses to zero-shot prompts that are comparable to those generated after supervised learning. Based on the number of labeled demonstrations, it can be categorized into zero-shot prompting with no examples and few-shot prompting with few examples.

Zero-shot prompting. Zero-shot prompting does not involve adding a labeled example to the prompt. It is composed of task and profile information [65]. This approach leverages the pre-existing knowledge of LLMs to generate responses based on the instructions of the task. Previous research [60] has shown that zero-shot prompting enables the model to access its existing knowledge by identifying already learned tasks. Furthermore, Reynolds et al. [60] suggested that there is significant potential for developing automated methods to generate task-appropriate zero-shot prompts.

Few-shot prompting. Unlike zero-shot prompting, few-shot prompting equips LLMs with a limited set of input-output examples, as illustrated in Fig. 4(a). This approach aids the model in comprehending both the task intent and the required output format. Providing several high-quality examples can improve the model's performance on complex tasks and standardize the form of outputs [66]. Carefully designed demonstrations within the prompt can achieve results comparable to fine-tuning, with the performance gap narrowing as the number of model parameters increases [13]. Several approaches are proposed for selecting and augmenting these demonstrations. For example, Liu et al. [67] and Su et al. [68] enhanced performance by choosing examples similar to the query input. Specifically, Liu et al. [67] employed a *K*-nearest neighbor

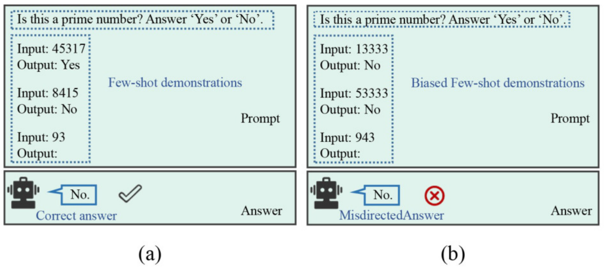

Fig. 4 The biased demonstrations direct the LLM to the wrong location of the knowledge. This causes LLM to over-reference incomplete examples. (a) Few-shot prompting; (b) misdirected few-shot

(*K*NN) approach, while Su et al. [68] introduced Vote-*K* to incorporate diverse and representative examples by artificially labeling useful, previously unlabeled examples. Additionally, Jiang et al. [69] optimized the structure of examples, moving beyond the conventional "Q&A" format to identify the most suitable prompt template for each query through large corpus analysis.

However, few-shot prompting has some limitations, as shown in Fig. 4(b). Few-shot prompting requires additional tokens to incorporate the demonstrations, which presents a limitation for processing long text inputs. Recent research [43,70,71] demonstrates that the selection and variation of samples can significantly impact model performance. In response to these findings, Fei et al. [72] introduced a systematic method for measuring label biases, identifying three distinct types of label biases in in-context learning (ICL) for text classification. To address these biases, several approaches [71,73] are proposed to calibrate the model's output probabilities. These methods typically utilize output probabilities generated from a set of inputs either sourced from the task domain [73,74] or from standard task inputs [75]. By adjusting the samples in few-shot learning, these techniques aim to produce unbiased outputs.

3.2 Knowledge

Insight: Knowledge outlines prompt engineering techniques based on Retrieval-Augmented Generation (RAG).RAG functions by retrieving relevant informationfrom a knowledge database based on an initial prompt and integrating it with the original prompt. This approach enhances the timeliness and professionalism of LLMs, effectively mitigating hallucination issues and improving the model's performance in specialized tasks.

Knowledge techniques involve augmenting prompts with external documents or knowledge databases. This process integrates content from local knowledge databases into the original prompt, enhancing alignment with specific information and addressing challenges such as hallucination and timeliness in LLM training [76–78]. As the scale of parameters of LLMs grows, the computing resources required for fine-tuning also increase. Retrieval-Augmented Generation (RAG) is a key technology for supplementing LLMs with real-world information, enabling them to fully utilize their reasoning capabilities. Although RAG techniques encompass both retrieval and knowledge graph methodologies [76,79], this paper primarily focuses on summarizing RAG techniques that are directly related to prompt design, while excluding those unrelated to prompt engineering. According to the complexity of the RAG framework, we classify it into basic RAG techniques and advanced RAG techniques.

1) Basic RAG techniques

Basic Retrieval-Augmented Generation (RAG) techniques dynamically retrieve information from external knowledge sources, organize the final prompt based on the original query, and use the retrieved data as a reference. The RAG workflow typically consists of three main steps [26], as illustrated in Fig. 5:

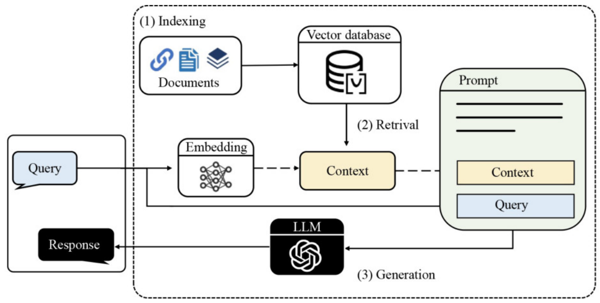

Fig. 5 A representative instance of the RAG process applied to question answering

- Indexing. Documents are divided into smaller chunks, encoded into vector representations, and stored in a database, such as an inverted index or a vector database.
- Retrieval. The top k chunks most relevant to the query are retrieved based on semantic similarity.
- Generation. The original query and the retrieved chunks are fed into a large language model (LLM) to generate the final response.

A simple RAG prompt might look like: "Please answer the above question based on: Segment 1: Segment 2:". The segment chunks are then populated with retrieved results from the external knowledge source. Basic RAG techniques provide a foundational approach to augmenting LLMs with external knowledge. This can be implemented using functions and other relevant tools. There are several potential areas for optimization within basic RAG techniques [80]. From a prompt engineering perspective, three key challenges are: 1) retrieving the most relevant document, 2) effectively combining the retrieved content to achieve optimal results, and 3) iteratively refining the entire process [80].

2) Advanced RAG techniques

In this part, we begin with an overview of the representative RAG framework and then explore optimisation methods from three distinct perspectives: pre-retrieval optimization, post-retrieval optimization, and memory management. As illustrated in Fig. 6, advanced RAG enhances the alignment between retrieved knowledge and the prompt, enabling the construction of more coherent and effective prompts. The advanced RAG prompt engineering enhances the workflow of basic RAG by incorporating several key optimizations: In the pre-retrieval stage, it addresses semantic discrepancies between the query and document chunks by rewriting and decomposing the query. In the post-retrieval stage, it refines results by modifying the content and structure of the "query+document". Additionally, advanced RAG leverages retrieval history and conversation history to generate memory for LLMs, improving the accuracy of responses in long conversations.

- Pre-retrieval optimization. The pre-retrieval optimization focuses on augmenting the original query in order to retrieve documents that are more relevant to the task. MultiHop-RAG [23] developed a dataset designed to strengthen RAG's capabilities. This dataset includes a knowledge base and multi-hop queries. The queries are categorized into inference, comparison, temporal, and null queries, each tailored to specific types of reasoning tasks. Query standardization is performed based on their unique characteristics, ensuring consistency and effectiveness across all four query types.

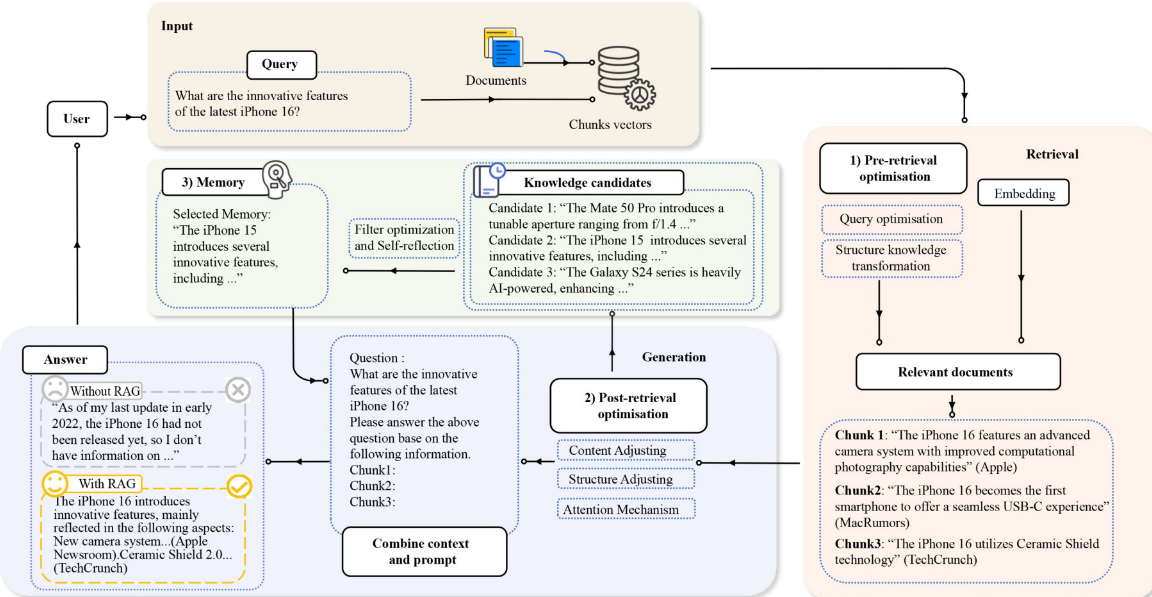

Fig. 6 A workflow of advanced retrieval-augmented generation (RAG) techniques. 1) Pre-retrieval optimisation phase. Before embedding the query, techniques such as query rewriting are employed to improve both the structure and content of the original query, facilitating the retrieval of more contextually relevant information. 2) Post-retrieval optimisation phase. Before inputting the retrieved documents into the LLM, the documents are refined through content filtering and structural adjustments to better align with the original query. 3) Memory module. This component preserves essential information and retrieval results in long-context dialogues, which enables more relevant retrieval when handling queries related to previous dialogue history

Query rewriting is another technique employed in pre-retrieval optimization. Query2doc [24] and ITER-RETGEN [81] both illustrate the effectiveness of query rewriting. These approaches leverage the capabilities of LLMs to generate pseudo-documents, which are then combined with the original query to form a revised version. This process effectively integrates corpus semantics into the user query. Query2doc has demonstrated significant improvements in BM25 [82] performance, achieving a 3% relative gain on MSMARCO [83] and a 15% relative gain on TREC DL [84], without requiring model fine-tuning. Similarly, ITER-RETGEN has shown enhanced performance across various question-answering tasks, including Natural Questions, TriviaQA [85], 2WikiMultiHopQA [86], and HotpotQA [87], surpassing previous baseline results.

● Post-retrieval optimization. The post-retrieval optimization focuses on the combination of retrieved documents and the user's original input. This approach refines the retrieved documents by adjusting their content, structure, and attention mechanisms as required.

(i) Content adjusting. RALM with CoN [88] reconstructs document content to enhance alignment with the original query. This method strategically emphasizes critical sections and modifies the retrieved document accordingly. Prior to inputting into the LLM, potential responses are generated from the retrieved content and subsequently evaluated to identify the most relevant context. ARM-RAG [89] takes a different approach, utilizing neural information retrieval to trace reasoning chains, particularly in solving mathematical problems. During their experiments, they found that accuracy could be improved by replacing words that might cause significant shifts in the model's reasoning. This technique involves blurring words that could disrupt the reasoning process, thus improving the model's overall accuracy.

(ii) Structure adjusting. Adjusting the structure of retrieved documents can enhance the effectiveness of the model's responses [90]. Articles can be classified into four categories based on their relevance to the query: Gold Documents, Relevant Documents, Related Documents, and Irrelevant Documents, in decreasing order of relevance [90]. Their study demonstrated that incorporating Irrelevant Documents into the document reconstruction process improves performance accuracy by more than 30%.

(iii) Attention mechanism. System2Attention (S2A) [91] enhances the soft attention mechanism in LLMs within the Transformer [92] architecture by refining and refocusing the attention process. Their approach leverages the LLM itself to build stronger attention mechanisms. Specifically, it uses prompts to adjust the LLM, enabling it to reconstruct the retrieved document into a new one by removing irrelevant text. Additionally, S2A introduces further techniques to refine attention. It generates the final response based on the reconstructed document, essentially refocusing attention one more time. S2A demonstrates promising performance, particularly on the TriviQA [85] dataset, where it outperforms LLaMA 2-70B-chat [93] in factuality with scores of 80.3% compared with 62.8%. On GSM-IC [94], S2A improves accuracy from 51.7% to 61.3%.

● Memory. In addition to constructing an effective external knowledge base, RAG can also store retrieved results and conversational histories to build memories. These memories are classified into two types: external and internal memories.

(i) External memories. The Selfmem framework [25] employs RAG in an iterative manner to create both a memory pool and a memory selector. This selector identifies an output to serve as memory for subsequent generation rounds. The core idea of this framework is based on prompt engineering. In this approach, the prompts presented to the LLM are crafted to be more similar to the model's outputs stored in the memory pool rather than the original training data. The memory pool undergoes multiple rounds of search optimization and retention to ensure that the most representative results are identified and preserved.

(ii) Internal memories. Internal Memories leverage the reasoning capabilities of LLMs to evaluate and provide feedback on the retrieved information, forming the internal memory of the LLM [26]. To implement this, the MetaRAG [95] framework is introduced, which incorporates three core processes: monitoring, evaluation, and planning during inference. MetaRAG has demonstrated significant performance improvements, achieving a 34.6% accuracy increase on the 2WikiMultihopQA dataset and a 26% accuracy improvement on the HotpotQA dataset.

3.3 Reasoning and planning

Insight: Reasoning and planning summarizes prompt engineering techniques aimed at enhancing the reasoning capabilities of LLMs, such as Chain-of-Thought (CoT) prompting. By decomposing tasks, leveraging external tools for reasoning, and incorporating feedback, it improves the model's ability to solve complex problems, as shown in Fig. 7. It represents a crucial step in prompt engineering and is a core component in the design of effective prompts.

1) Target decomposition reasoning

Target decomposition is the key prompt technology that enhances the reasoning ability of LLMs. It mirrors the core of human reasoning, where individuals, through experience, learn to tackle complex goals by breaking them down into more manageable sub-goals [96]. This section introduces the essential prompt strategies: plan-execute decomposition and iterative decomposition, as shown in Fig. 7.

● Plan-execute decomposition. The approach involves decomposing a complex query Q into a series of simpler subproblems that can be addressed sequentially. This method emphasizes the relationship between problem-solving steps, enabling the inheritance and promotion of previous solutions. The Least-To-Most Prompting method [97] consists of two main phases: Decomposition and Sub-problem Solving. In the Decomposition phase, the prompt includes examples and specific instructions to illustrate the breakdown process. Subsequently, in the Sub-problem Solving phase, attention shifts to demonstrating how each sub-problem is resolved using the provided examples, which are recorded in a list. This list stores previously answered sub-questions along with their corresponding answers, and it helps identify the next question to address in the sequence.

● Iterative decomposition. Chain of Thought (CoT) is the core technique of iterative decomposition. CoT decomposes task goals during the reasoning process. It iteratively decomposes the problem to identify and address sub-goals. This process is then repeated until the target is fully completed. Finally, the reasoning results are generated. The origin of the CoT within LLMs can be linked to the pioneering ideas proposed by Jason et al. [30]. Their ideas revolve around generating thought chains to enhance LLMs' ability to perform complex reasoning tasks. When LLMs are provided with limited samples during inference, the prompt follows a structured triplet format: <input, chain of thought, output>. This structured framework equips LLMs with the capacity to produce similar chains of reasoning, offering valuable insights into their computational capabilities and reasoning processes.

(i) User-level CoT prompt. The work by Wei et al. [30] is regarded as the pioneering study on CoT prompting. By providing the LLM with a sequence of intermediate reasoning steps within the prompt, the model can emulate the human problem-solving process, ultimately arriving at an accurate solution. This process can be further simplified through the use of zero-shot and few-shot prompting techniques, as outlined in Subsection 3.1. Additionally, Kojima et al. [64] demonstrated that the inclusion of the phrase "Let's think step by step" in the prompt enhances the LLM's ability

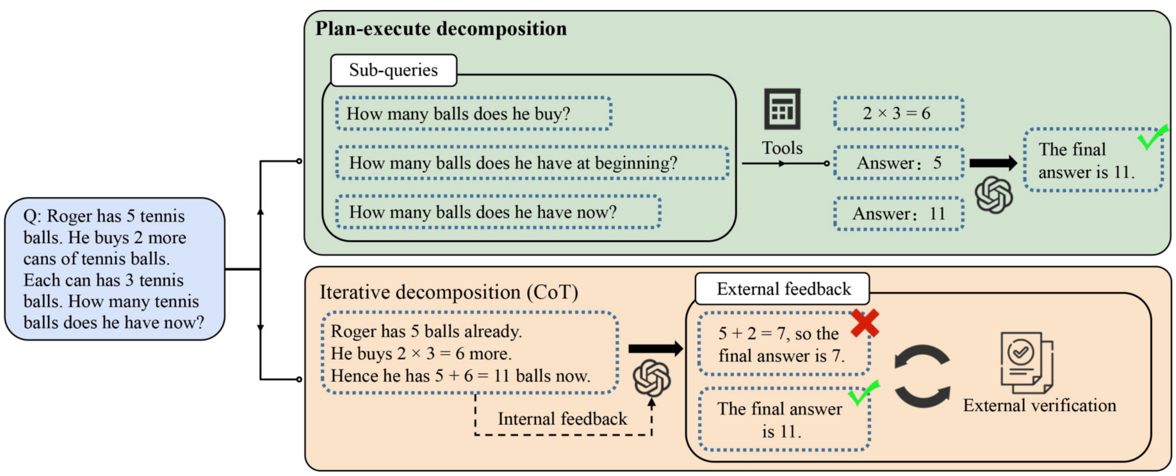

Fig. 7 An example of utilizing reasoning techniques to solve problems. Task decomposition methods complement LLMs by structuring the reasoning process, thereby enhancing their ability to tackle complex problems. The integration of tool utilization within prompts enables LLMs to concentrate on reasoning tasks, while feedback mechanisms further strengthen their reasoning capabilities

to perform CoT decomposition. Leveraging few-shot examples, LLMs can autonomously decompose and solve complex problems, leading to reductions in both energy consumption and processing time. The scope of the CoT is extensive, encompassing a wide range of problems encountered in daily human life, including mathematical calculations, common-sense reasoning, and more. Notably, CoT, based on few-shot examples, demands less effort and time compared to directly training an LLM.

(ii) Technique enhanced CoT. Following the introduction of CoT, researchers have focused on improving its efficiency and adaptability.

Auto-CoT automates problem sampling and inference chain generation using Question Clustering and Demonstration Sampling [98]. In clustering, Sentence-BERT [99] encodes queries as vectors, grouped via K-Means to form clusters of related problems. Each cluster's central problem is selected for Zero-Shot-CoT [64], generating structured inference chains. Active Prompt [100] addresses the challenge of adapting CoT manual annotation examples to diverse tasks. It enables LLMs to adjust to task-specific requirements through example prompts annotated with manually designed CoT inferences. To improve CoT interpretability, Faithful CoT [101] integrates Natural Language (NL) and Symbolic Logic (SL) programs for structured reasoning. The NL program decomposes queries into subproblems, while the SL program (e.g., Python and Datalog) solves them iteratively, enhancing accuracy and explainability.

Researchers have extended Chain-of-Thought reasoning by developing more flexible frameworks, among which XoT [102] represents a significant advancement. This approach enables adaptive switching between reasoning modes when encountering obstacles, guided by two distinct validation methods. Passive validation ensures fundamental correctness through basic error detection, while active validation assesses whether generated responses align with the original query. XoT also introduces diverse structural formats to enhance reasoning capabilities. Tree-of-Thought [102] structures reasoning hierarchically, enabling step-by-step decomposition, whereas Graph-of-Thought [32] models reasoning as an interconnected network, allowing for greater flexibility and adaptability in complex problem-solving.

2) Tools and feedback enhanced reasoning

Advanced reasoning techniques utilize systematic methodologies to optimize prompt design. These techniques include delegating specific computational tasks to external tools and simulating human feedback learning processes. The following discussion is organized into two sections: external support and feedback.

● External support. LLMs are expected to perform both semantic understanding tasks (e.g., task intent understanding) and complex reasoning tasks (e.g., numerical computation) when used for reasoning. However, current LLMs often face challenges in executing both tasks simultaneously. For instance, in complex code generation tasks, the generated code frequently contains errors or bugs. External support reduces the workload of LLMs by utilizing tools, allowing them to focus on reasoning and planning. According to the external support they utilize, external support is categorized into experimental simulators, code interpreters, and integration tools.

(i) Experimental simulator. Current LLMs face challenges when dealing with real-world problems that require a deep understanding of physical principles [103]. Due to the limitations of LLMs, LLMs' responses are often based on the semantic interpretation of the text rather than the rules of physics, making it difficult for them to reason logically through existing knowledge [104,105]. As a result, relying solely on LLMs for solving physical problems is problematic. To address this limitation, it is crucial to conduct simulation experiments on physical problems and incorporate the results into LLM-generated responses.

Mind's Eye [103] leverages a computational physics engine [106] to simulate real-world physical processes. A Text-to-Code language model is employed to generate rendering code for the physics engine, allowing the simulation of physical experiments relevant to the posed question. The simulation outcomes are then incorporated into the LLM prompt in natural language, compensating for the model's lack of physical understanding. Mind's Eye demonstrates a significant improvement in inference accuracy.

(ii) Code interpreter. When confronted with complex reasoning problems, offloading the precise computation task to external modules can significantly improve the model's solution accuracy. Researchers [107,108] prompt Codex [50] to generate executable, code-based solutions. These solutions address a variety of tasks, from university-level exercises to mathematical word problems and financial question-answering (QA). This idea of cooperation between LLMs and code has also been applied to solve more specific problems [109,110]. One possible interpretation of these findings is that code-based approaches benefit from well-defined structural consistency, offering advantages in robustness and logical reasoning compared to natural language. This form of code-driven reasoning allows LLMs to focus more on problem-solving logic rather than the intricacies of textual representation.

The PAL [111] method employs a CoT prompting strategy to decompose complex symbolic reasoning, mathematical problem-solving, and algorithmic tasks into intermediate steps, represented through Python code and natural language annotations. In this setup, computational tasks are delegated to the Python interpreter. Similarly, Chen et al. [112] introduced "Program of Thoughts" (PoT) prompting, which separates computational and reasoning processes through specific prompt designs. Unlike PAL, PoT follows a zero-shot approach for prompt generation. Luo et al. [113] proposed a framework named MultiPoT to select the optimal external programming language based on task types to overcome the Python language's extension limitations. MultiPoT creates custom hints for each programming language to ensure semantic consistency and structural diversity. When addressing issues, it integrates multiple programming languages and selects the final answer generated from each PL by self-consistency. The results obtained incorporate the benefits and diversity of multiple programming languages.

(iii) Integration tools. In addition to code interpreters and simulators, more tools have been collected in the form of APIs. Integration tools are the frameworks that incorporate diverse tools into prompts. The strong generalization ability of LLM allows it to effectively utilize natural language as an intermediary to manipulate external tools [114]. Additionally, prompt engineering techniques serve as the guides for these tools, providing instructions and guidance on how to effectively utilize them. Based on this insight, Parisi et al. [115] built a text-to-text_api prompt enhancement framework called TALM. The TALM bootstrap LLM generates "|tool-call" and "tool input text" to construct API call requests, and then appends the results returned by external APIs to the text sequence, significantly expanding the model's capabilities. Similar to this, Toolformer [116] uses the same approach of including API requests in the prompts to leave the sub-operations to external tools, while Toolformer also suggests ways to build datasets and fine-tune them. Galactica [117] adopts a special token to initiate a request to call an external tool. This work delivers the concept of "work memory", which generates a special segment in which algorithms and problem-solving code are generated when the model needs to call an external tool. These methods require a strategy to determine what tasks should be offloaded.

To overcome the limitations of manually writing API requests, some works provide strategies to make LLM use tools automatically. MRKL [118] provides a modular neural-symbolic architecture that divides tasks into corresponding task APIs through a router and integrates the returned results. ART [119] organizes its task library, retrieves tasks related to the original prompt in the task library, then decomposes a series of tasks in turn into corresponding sub-tool sequences, generates a specific cue word to be processed, and finally integrates the results of the tool API into the original cue word to realize the automated tool usage of LLM. More large-scale architectures such as Chameleon [120], Gorilla [121], HuggingGPT [122], and ToolAlpaca [123] take advantage of richer API tools and more systematic API requests, greatly enhancing the ability of LLM to use tools to handle tasks. Based on these architectures, several "LLM + Tool" agents such as TPTU [124], TPTUv2 [125], and TaskMatrix. AI [126] have also demonstrated a strong ability to handle complex tasks in different scenarios.

● Feedback. Similar to human learning processes, feedback enables individuals to identify areas for improvement and resolve issues more effectively. The provision of feedback enables the LLM to ascertain the correct and incorrect responses through prompting, thereby enhancing its capacity for reasoning. In accordance with the subject of the judgment, feedback can be classified into two distinct categories: internal feedback and external feedback.

(i) Internal feedback. LLMs possess the inherent capability to engage in self-feedback. Numerous scholars in this field have demonstrated the self-feedback ability of LLMs and have proposed corresponding methods to enhance and leverage this mechanism effectively. By harnessing self-feedback, LLMs can iteratively refine their outputs, contributing to their continual improvement and better performance.

Internal feedback techniques. Some scholars [127] demonstrated that LLMs can undergo iterative self-refinement without requiring additional training, introducing the SELF-REFINE approach. This method refines the output of LLMs through alternating iterations of feedback and refinement to enhance the output quality. The feedback generation process is guided by a few-shot prompt [13]. SELF-REFINE outperforms models such as GPT-3.5 and GPT-4, directly yielding absolute improvements ranging from 5% to 40%. In tasks involving code generation, when applied to CODEX, SELF-REFINE improves initial generation by up to 13%.

SELF (Self-Evolution with Language Feedback) [128] enables LLMs to engage in self-feedback and self-refinement. This is achieved by providing LLMs with feedback in natural language, empowering them to autonomously evolve. The evolution process involves generating responses to unlabeled instructions and iteratively refining these responses through interactions. SELF teaches LLMs fundamental meta-skills using a limited set of examples in natural language, fostering a continuous cycle of self-evolution for LLMs. Self-Contrast [129] enhances the self-feedback capacity of LLMs. This approach prompts LLMs to generate various perspectives to address the same problem, subsequently facilitating comparison and reflection on the disparities among these perspectives.

Some scholars [130] proposed that LLMs can autonomously provide self-feedback regarding hyperparameter perception. Through hyperparameter-aware instruction tuning, LLMs ascertain the optimal decoding strategy and configuration based on input samples, thereby achieving self-regulation. Their proposed HAG (Hyperparameter Aware Generation) consists of two components. Firstly, by inputting a query Q to the LLM, HAG generates appropriate hyperparameters. Subsequently, HAG instructs the LLM to adjust the model's decoding strategies and hyperparameters according to the generated parameters, culminating in the generation of the final result post-adjustment.

Evaluation of internal feedback. For the evaluation of internal feedback, some scholars [131] proposed that glass-box features should be taken into account in self-assessment of LLMs. They proposed that the softmax distribution serves as a dependable indicator for quality evaluation. Self-feedback can enhance performance on certain tasks for LLMs, while potentially worsening performance on others [132]. In light of this, some scholars [133] introduced the concept of Self-Bias to evaluate the bias of LLM output accuracy across different tasks. Additionally, their research reveals that models with larger parameters and access to external feedback possess the ability to more accurately assess and mitigate Self-Bias.

(ii) External feedback. Certain external tools can verify the extrapolation outputs of LLMs. CRITIC framework [134] enables appropriate tools to assess the output of LLMs, which in turn allows LLMs to adjust and refine their output based on this feedback. These external evaluation tools encompass search engines, code interpreters, text APIs, and more. The feedback process primarily involves validating the initial output expectations generated by the LLMs and then modifying the output based on critiques from the verification process. Through this iterative process, external tools offer feedback to LLMs from an external perspective, thereby enhancing the performance of LLMs.

3.4 Reliability

Insight: Reliability, as the final step in the prompt design process, focuses on ensuring more consistent responses from LLMs while reducing biased outputs. It provides a foundation for the practical deployment of LLMs in realworld applications.

In the first three parts, we introduce how to design an effective prompt for LLMs using the Profile-Knowledge-Reasoning process. Due to the uncertainty of the LLM's response [135], the same prompt can lead to diversified responses from the LLM. At the same time, due to the LLM's context learning ability, it is particularly sensitive to the order, type, and historical bias of the prompts [135–137]. The bias of the prompt may worsen the performance of the LLM in downstream tasks. Furthermore, the reasoning strategy used by the LLM is opaque, which means that the responses of the LLM are not always trustworthy [138], and there are a series of risks associated with the responses of the LLM [139,140]. In practical applications, users often demand stable and reliable responses from the LLM. The process of reducing the bias of the LLM's response through prompt integration or by generating auxiliary knowledge from the LLM is known as improving the reliability of the LLM. In this part, we will explain from both perspectives of content bias and value bias.

1) Content bias

We define content bias as the deviation between the completion results of the LLM and the task requirements. For most LLM tasks, designing prompts according to the profile-knowledge-reasoning process can help the LLM perform well in response to task requirements. However, when the task is more complex, it is difficult to obtain stable and perfect output through a single round of prompt input [137,139]. An intuitive idea is to adjust the order of the prompts or change the method of prompting to generate multiple rounds for the same question, which actually uses the concept of ensemble. The prompt method based on this idea is called prompt ensembling.

Prompt ensembling refers to the use of multiple different prompts to complete the same task, enhancing the reliability of the results through multiple responses. Prompt ensembling borrows from the pattern of ensemble learning and includes two sub-processes: first generating multiple prompts as input, and then combining the responses of multiple prompts through a specific strategy to obtain the final result [141,142]. Bagging and boosting are two typical ensemble methods widely used in many classic tasks and have unique applications in LLM.

● Bagging prompt. The application of Bagging Prompt in LLM mainly falls into two categories: the majority vote method based on Self-Consistency [143] and the beam search method based on Step-Verifier [144]. BPE [145] focuses on constructing few-shot CoT prompts based on Self-Consistency, which outperforms a single prompt. However, since Self-Consistency is a method based on a greedy approach, it cannot guarantee that the inference chain is entirely correct. Additionally, its voting mechanism is atomic, meaning it lacks the ability to differentiate the quality among various responses.

In response to these challenges, DiVeRSe [146] was developed to enhance answer reliability through a three-stage process: "generate-verify-check". First, it generates diverse completions using multiple prompts. Next, a "step-aware voting verifier" model distinguishes good answers from poor ones and verifies the correctness of inference steps. DiVeRSe extends traditional methods by extracting step-level labels from intermediate results, ensuring accuracy in the inference flow. While it integrates the benefits of bagging Prompt methods, DiVeRSe still relies on manual sample selection, and sample bias can affect final results.

Motivated by these issues, the AMA (Ask Me Anything) method was introduced [147]. AMA consists of two stages: the multiple prompt step and the answer aggregation step. In the multiple prompt step, AMA employs a functional prompt chain where the question() function transforms the input into open-ended questions, providing diverse perspectives for LLMs to address different aspects of the problem. The answer() function then generates intermediate answers. AMA highlights the limitations of simple voting due to equal weighting and question similarity, which can distort results. To overcome this, AMA uses an answer aggregation method based on weakly supervised learning with an information entropy penalty, ensuring that results better reflect different perspectives. Essentially, AMA optimizes the Bagging random sampling approach.

● Boosting prompt. Boosting prompt methods often adopt a two-stage paradigm. PromptBoosting [148] applies the AdaBoost algorithm on the prompt set, achieving good results in text classification. However, the PromptBoosting method requires a prepared high-quality prompt set and cannot optimize for specific prompts.

To overcome these limitations, Prefer [149] establishes a feedback mechanism to reflect on the shortcomings of the weak learners in the current iteration. Based on feedback, Prefer also implements the automatic synthesis and selection of prompts, avoiding the bias problem brought by the prompts. PromptBoosting's ensemble method for weak learners refers to the traditional ensemble method of weighted summation. However, many works have pointed out that LLMs have a serious optimistic estimation problem [150,151]. This makes the weighted calculation of answer accuracy unable to eliminate content bias.

Bilateral Prompt Bagging [149] assesses the confidence of the generated results in each iteration. When the assessment result of the answer is not trustworthy, a reverse confidence evaluation is performed, calculating the confidence that this answer is incorrect. The final correct probability of a round of generated results is evaluated by combining forward and reverse confidence. This design of positive and negative confidence effectively avoids the optimistic estimation problem of LLMs [150]. By leveraging the feedback reasoning capabilities of LLMs (as discussed in Subsection 2.3), it enhances the quality of single-round generated results, strengthens weak learners, and ensures a more reliable and efficient final outcome.

2) Value bias

We define value bias as the deviation between the content generated by LLMs and human societal values. The content generated by LLMs often touches upon socially sensitive areas, such as issues of social harm and discrimination [152,153]. Shaikh et al. [139] showed that Chain of Thought (CoT) can continually enhance the performance of LLMs across a variety of NLP tasks. However, this also increases the likelihood of the model producing harmful or inappropriate results. Research [154] indicates that CoT prompts should be used carefully when dealing with socially relevant issues. Due to the principles of autoregressive decoding inherent in LLMs, harmful text might still be generated during the beam-search phase of the inference stage if the volume of harmful text in the training corpus is large enough. This underlines the importance of developing mechanisms to manage and minimize harmful or socially unacceptable outputs for the safe deployment of AI systems.

Tang et al. [155] proposed the Detox-Chain method, which links detoxification sub-steps together to achieve rapid detoxification of prompts. The Detox-Chain method reduces the possibility of LLM generating toxic texts by substituting toxic textual representations. However, due to the detoxification technology for prompts altering the distribution or content of the prompt to varying extents, it affects the quality of the output produced by the LLM. At the current stage, the detoxification methods for LLMs are still primarily based on reinforcement learning or knowledge editing techniques during the training phase.

4 Applications

This section offers practitioners a comprehensive overview of current application domains for LLMs and highlights the prompt engineering techniques utilized across diverse use cases. While existing studies [156] have introduced taxonomies for LLM applications, these classifications tend to focus on specific scenarios, limiting their relevance for prompt engineering design. To address this gap, we propose a novel taxonomy that categorizes applications by task characteristics into two primary types: cognitive applications and transformative applications of LLMs. Cognitive applications highlight LLMs' roles in providing and processing information, whereas transformative applications involve tasks where LLMs operate autonomously, potentially substituting human involvement. This taxonomy provides detailed insights into prompt engineering techniques tailored for each application category, as illustrated in Fig. 1.

4.1 Cognitive applications of LLMs

Cognitive applications of LLMs involve utilizing these models to acquire and analyze information. The process of leveraging LLMs to extract and present knowledge is termed Information Acquisition. This category encompasses applications such as chatbots, professional knowledge Q&A, search engines, and training data augmentation. Conversely, when LLMs are employed to interpret and analyze input data, the process is referred to as In-depth Information Analysis. Key applications in this analytical domain include the simulation of financial market dynamics and the exploration of chemical molecular structures, showcasing the model's ability to provide insightful analyses and predictive insights across specialized fields.

1) Information acquisition

Information acquisition is a fundamental application of LLMs, leveraging their extensive knowledge base to efficiently deliver relevant information to users.

> **Finding:** Information acquisition is supported by various prompt engineering techniques, which deliver the required information to users. *Profile* techniques equip the LLM with detailed, task-specific prompt, enabling a more precise context. *Knowledge* techniques provide the model with up-to-date, task-relevant information, helping prevent overconfident responses. When addressing specific queries, reasoning techniques enhance the model's problem-solving capacity, significantly improving its ability to generate accurate and reliable answers.

● Chatbots. General chatbots, also known as dialogue agents, integrate tasks such as information retrieval, multi-turn interaction, and text generation (including code). LLMs encapsulate extensive knowledge within their parameters during training. Users can obtain information from LLMs through role-playing dialogue, exemplified by the chatbot [157].

For instance, Glaese et al. [158] introduced Sparrow, a dialogue agent based on a 70B parameter Chinchilla LLM. It utilizes prompt engineering techniques, including Basic RAG Techniques (Subsection 3.2) and Personality Information (Subsection 3.1), to address hallucination issues by integrating external knowledge from Google search queries.

Similarly, OpenAI [159] utilizes supervised fine-tuning with high-quality data, along with reinforcement learning from human feedback (RLHF), to develop the GPT-3.5 LLM, which powers the ChatGPT chatbot. The subsequent GPT-4 model [160] underpins the ChatGPT Plus chatbot. ChatGPT employs prompt techniques such as Personality Information (Subsection 3.1) for role-playing conversations and Memory (Subsection 3.2) to maintain conversation history, thereby excelling in extended discussions. Microsoft Copilot [161] is an AI-powered productivity tool that integrates an LLM with Microsoft 365. Supported by an OpenAI model, Copilot utilizes additional integration tools (Subsection 3.3), accesses up-to-date information via Bing, and incorporates Retrieval-Augmented Generation (RAG) technology (Subsection 3.2) to enhance user prompts.

Anthropic [162] introduces the Claude series of chatbots. These bots are further refined via fine-tuning with high-quality data and guided by RLHF to generate responses that are beneficial, harmless, and honest. The Claude 3.5 Sonnet model has outperformed competitor models in various AI system evaluation benchmarks, including undergraduate-level expert knowledge (MMLU), graduate-level expert reasoning (GPQA), and basic mathematics (GSM8K).

● Search engines. Information Retrieval (IR) systems are integral to dialogues, question answering, and recommendation systems, serving as primary means of obtaining information. LLMs can improve traditional IR components, such as query rewriters, and can also function as the engine for generative retrieval.

(i) LLM-based query rewriter. The query rewriter enhances user queries by adding synonyms or related terms to address vocabulary mismatches and clarify ambiguities, thus aligning more accurately with user intent. In conversational retrieval, the query rewriter comprehends the context of the entire conversation, clarifying ambiguous content and generating a more effective new query based on the user's dialogue history.

HyDE [163] represents pioneering work on LLM-based query rewriting, guiding LLM generation using various prompt engineering techniques in Profile and Perception (Subsection 3.1). HyDE generates detailed hypothetical documents based on the given query, which are then retrieved from the corpus using a dense retriever. Query2doc [24,164] exemplifies another innovative approach to query rewriting, generating pseudo-documents by prompting LLMs with a few demonstrations, reflecting the use of the Demonstration Information approach (Subsection 3.1). These pseudo-documents are subsequently expanded with generated documents. Based on prompt engineering optimization, Jagerman et al. [165] studied the impact of different prompting methods and model sizes on query rewriting [165].

GFF [166] adopts a "generate, filter, and fuse" method for query expansion, utilizing LLMs to extract related keywords from the original query through a reasoning chain. GFF employs prompt ensembling strategies (Subsection 3.4) to filter generated keywords using techniques such as Self-Consistency [143], ensuring the quality and relevance of keywords, which are then integrated with the original query for downstream reranking tasks.

(ii) LLM-based generative retrieval. Generative retrieval methods utilize a unified model to directly generate document identifiers (DocIDs) related to user queries, integrating prompt engineering techniques. Traditional IR systems typically follow the "index-retrieve-rerank" paradigm, which has proven effective in practice [163,167]. However, the constituent modules—indexing, retrieval, and reranking—operate independently. LLM-based generative retrieval unifies these modules, initiating a new paradigm for IR systems.

Researchers have demonstrated that LLMs, such as the GPT series, can directly generate URLs related to user queries [168]. This capability allows the LLM to function as a generative retriever, producing document identifiers from the original input to obtain relevant documents. Ziems et al. [168] introduced the LLM-URL model, which employs the GPT-3 text-davinci-003 model to generate candidate URLs. It utilizes the Demonstration Information approach (Section 3.1) to standardize model output and incorporates a URL filtering mechanism that extracts valid URLs from candidates using regular expressions.

● Professional knowledge Q&A. The expansive capabilities of LLMs enable them to possess specialized knowledge across various fields, effectively organizing and presenting information to address specific inquiries. This ability to provide accurate responses to user queries is commonly referred to as professional question and answer (QA).

LawGPT [169] is a robust language model designed to address legal knowledge inquiries with high accuracy. It employs the LLM internal feedback method (Subsection 3.3) to continually enhance its precision in legal question answering. Through this approach, LawGPT generates legal queries pertinent to specific legal texts and subsequently provides responses in the form of "text segment-question" pairs, ensuring that its answers are rich in legal information.

MultiMedQA [170] is a benchmark that integrates clinical expertise with medical knowledge, employing a Few-shot prompting technique (Subsection 3.1). Collaborating closely with a panel of seasoned clinicians, MultiMedQA produces exemplary few-shot demonstrations and sample scenarios. Moreover, it adeptly demonstrates Reasoning and Planning capabilities, exhibiting a robust chain-of-thought process (Subsection 3.3) for a myriad of medical issues by leveraging insights from various medical professionals. Furthermore, MultiMedQA excels in internal feedback (Subsection 3.3) verification, a crucial aspect in refining its accuracy. Given the multifaceted nature of medical queries, it employs the self-consistency strategy (Subsection 3.4), allowing the model to compare and evaluate diverse perspectives to ensure coherence and reliability in its responses.

● Training data augmentation. Due to the high cost of manually annotating labels, a common challenge in training neural retrieval models is the lack of training data. LLMs can learn patterns from manually annotated data and generate additional data consistent with the existing dataset.

Yoo et al. [171] proposed GPT3Mix, which generates synthetic data from existing datasets based on the GPT3 LLM. GPT3Mix employs the Demonstration Information method (Subsection 3.1) and the Task Information method (Subsection 3.1). The prompts of GPT3Mix include two parts: real examples from the dataset and task specifications, which are used to create synthetic data and pseudo labels.

Yoo et al. [171] utilized this new augmented dataset to fine-tune BERT and DistilBERT models, achieving excellent performance in classification tasks. In the context of information retrieval, it is easy to collect many documents; however, the challenging and expensive task is to collect real user queries and label the relevant documents accordingly. Given the LLM's powerful natural language processing capabilities, many researchers [172,173] suggested using an LLM-driven process to create pseudo queries or relevance labels based on existing datasets.

In cutting-edge work, Dai et al. [174] introduced ChatAug, which transforms training samples from a small dataset into multiple samples that are conceptually similar but semantically distinct. ChatAug outperforms state-of-the-art text data augmentation methods in terms of test accuracy and augmented sample distribution.

2) In-depth information analysis

In addition to providing information, LLMs can leverage their reasoning capabilities to analyze patterns within specific data or information. Applications requiring the evaluation of complex systems are categorized under In-depth Information Analysis.

Finding: Unlike information acquisition, in-depth information analysis presents LLMs with more complex challenges, such as financial market analysis. Profile techniques provide task information and demonstrations, enabling the LLM to better align with specialized requirements. By retrieving more specific task data and memorizing essential information, knowledge techniques further enhance the model's domain expertise. Employing professional tools to assist with task handling can effectively support the model in generating more valuable analysis.

● Financial markets. By providing Internet-scale data for LLMs, these models can utilize hybrid training methods in financial tasks, driving open-source development in the financial field. BloombergGPT [175] is a formidable large language model with 5 billion parameters, meticulously trained to excel in the intricate realm of finance through vast datasets.

BloombergGPT combines prompt engineering technologies such as personality information (Subsection 3.1) and external financial tools (Subsection 3.3) to provide personalized, time-sensitive financial advice. Its versatility extends across various tasks within the financial sector, making it beneficial for professionals. Leveraging techniques such as few-shot learning and other sophisticated methodologies, scholars fine-tune BloombergGPT to suit specific task formats, thereby enhancing its efficacy in delivering best-in-class results.

FinGPT [176] stands out as an open-source, large language model tailored specifically for the financial sector, boasting versatile applications such as robot consultation, algorithmic trading, and low-code accessibility. Its remarkable capability lies in making quantitative inferences from vast financial datasets, effectively capturing the intricate dynamics of the financial markets. A key feature of FinGPT is its adept utilization of external feedback methods (Subsection 3.3) within prompt engineering. By leveraging stock prices as indicators, FinGPT engages in reinforcement learning, continuously refining its understanding and interpretation of financial texts. This process empowers FinGPT to anticipate and predict market responses to a wide array of financial events, thereby enhancing its overall performance and utility in financial analysis and decision-making.

● Chemical molecules. The study of chemical molecules encompasses the properties, composition, structure, and chemical reactions of matter. This field involves complex computational and predictive tasks, such as property prediction, chemical structure optimization, and reaction prediction. As data scales increase, traditional chemical computational methods can no longer meet the demand, making the application of LLMs in chemistry increasingly important.

BioinspiredLLM [177] is an autoregressive transformation large language model specifically tailored for biomaterials and biomimetic materials. This sophisticated model excels in accurately recalling vast amounts of information pertaining to biomaterials, effectively reflecting intricate patterns between various biomaterials, and facilitating research tasks within this field. Scientific researchers continuously uncover new patterns, and BioinspiredLLM accommodates this by offering additional contextual content when queried. Moreover, the model leverages the Retrieve and Generate (RAG) framework, based on Knowledge prompt engineering techniques (Subsection 3.2), to provide comprehensive and insightful answers.

ChatDrug [178] harnesses conversational and reasoning capabilities to facilitate AI-assisted drug discovery endeavors. ChatDrug employs Chain-of-Thought (CoT) technology (Subsection 3.3) to decompose complex concepts into more understandable attributes, thereby improving reasoning ability and problem-solving efficiency. Additionally, ChatDrug incorporates prompt design tailored for domain-specific modules, with the flexibility to implement prompt functions via few-shot learning methodologies (Subsection 3.1). Leveraging the Retrieve and Generate (RAG) method (Subsection 3.2) enriches ChatDrug's professional knowledge, thereby enhancing the quality of responses.

ChemCrow [179] enables reasoning across common chemistry tasks such as material design and synthesis, from reasoning about simple drug discovery cycles to planning the synthesis of substances across a wide range of molecular complexity. ChemCrow combines the inferential power of LLMs with the chemistry expertise of computational tools (Subsection 3.2), successfully planning and synthesizing an insect repellent, three organocatalysts, and guiding the screening and synthesis of a novel chromophore with target properties. Enhanced by the RAG method (Subsection 3.2) and search engine tools (Subsection 3.3), ChemCrow can obtain scientific information from the Internet and build relevant databases. It can also accept external feedback from humans and adjust itself accordingly.

4.2 Transformative applications of LLMs

Transformative applications of LLMs facilitate automation and enhance workflows traditionally dependent on human input. These applications can be broadly categorized into physical world applications and creative applications. Through prompt engineering, LLMs can impact the physical world across various application scenarios, including software engineering, robotics, and expert-level task automation. Conversely, the impact of LLMs on creative domains extends to areas traditionally reserved for human creativity and artistic expression, including language generation, visual content creation, and auditory processing. These enhancements contribute to expanding human creativity and comprehension.

1) Physical world applications

Applications of LLMs in the physical world encompass areas such as software engineering, robotics, and advanced automation.

> **Finding:** The powerful capabilities of LLMs demonstrate their significant potential to replace repetitive manual human tasks. *Profile* techniques provide task-specific requirements and real-world environment information. In specialized domains, such as AI-driven healthcare, *knowledge* techniques supplement the model with domainspecific expertise, while enhancing reliability by providing professional knowledge. *Reasoning* techniques can effectively enhance the capabilities of LLMs in addressing a wide range of complex real-world problems. *Reliability* techniques ensure that the model's actions remain stable and dependable.

● Software engineering. One of the most advanced and widely used applications of LLMs is generating and completing computer programs in various programming languages. This section discusses programming-specific LLMs, which are fine-tuned or pre-trained specifically for programming applications. However, it is important to note that general chatbots, which are partially trained on code datasets (such as ChatGPT), are increasingly utilized in programming tasks.

Code generation refers to the use of LLMs to output new code based on the requirements provided in the prompts. Several LLMs and methods have been proposed for computer programming. OpenAI first released CodeX [50], an LLM based on GPT-3 (up to

12B parameters) pre-trained on public datasets to generate independent Python functions from natural language strings. CodeX standardizes output using prompt techniques such as personality information (Subsection 3.1) and few-shot prompting (Subsection 3.1).

The Copilot [161] plugin based on CodeX has also become a benchmark for code generation assistance tools. The HumanEval [50] dataset has become one of the commonly used benchmark datasets for subsequent code generation. Nijkamp et al. [51] sequentially trained CodeGen series LLMs (up to 16B parameters) on three datasets: the natural language dataset (THEPILE), the multilingual programming source code dataset (BIGQUERY), and the single-language Python dataset (BIGPYTHON). CodeGen models have been trained in various programming languages, including C, C++, Go, Java, JavaScript, and Python. The results indicate that the largest CodeGen model outperforms the Codex-12B model. CodeGen can utilize CoT technology (Subsection 3.3) to enhance the reasoning ability of generated code.

Nijkamp et al. [51] also tested CodeGen for multi-step program synthesis, decomposing the program into multi-step natural language prompts, with the synthesis system completing the synthesis of subroutines at each step. They also created a multi-round programming benchmark (MTPB) for the multi-round program synthesis method. Due to its excellent performance across multiple programming languages, Zheng et al. [180] trained CodeGeex on three datasets: The Pile, CodeParrot, and public repository supplement data on GitHub. CodeGeex employs a standard Transformer architecture, supporting high-precision code generation while also enabling automatic translation and conversion of code snippets between different programming languages.

● Robotics. Multimodal large language models are increasingly applied in robotics, where robots interact with the physical world through various methods. These models leverage their reasoning and planning capabilities to guide robot actions.

PaLM-E [52], a general visual language model, incorporates embedded data into multimodal training, serving as an effective inference engine. Experimental results demonstrate that, in addition to general visual language tasks, PaLM-E excels in tasks such as entity capture, performing admirably in real desktop and mobile manipulation scenarios. Fine-tuning of PaLM-E involves scaling the language model size, enhancing its adaptability across different tasks and environments. This multimodal approach ingeniously employs prompt engineering across diverse tasks and scenarios. Task information (Subsection 3.1) enables PaLM-E to discern the attributes of various tasks, while task environment information provides tailored scenarios for specific tasks. Additionally, PaLM-E utilizes Target decomposition reasoning (Subsection 3.3) to break down input tasks in advance, enabling the multimodal model to provide more realistic and effective responses.

● Expert-level task automation. In contrast to applications aimed at information retrieval, expert-level task automation involves applications where LLMs serve as substitutes for human experts in specialized fields. Examples include LLM-based agents that perform the functions of professionals, such as legal advisors or medical practitioners.

WisdomInterrogatory [55] is adept at responding to new legal issues using existing legal documents and provisions within the field of law. Through task information (Subsection 3.1), it can utilize corresponding legal knowledge to answer consultations. WisdomInterrogatory systematically clarifies attribute information, information sources, task types, and answer plans for various legal scenarios. It employs RAG technology (Subsection 3.2) for retrieving relevant legal information and enhances relevance through pre-retrieval optimization (Subsection 3.2). By prompting the LLM with few-shots (Subsection 3.1), WisdomInterrogatory can generate tailored responses for specific legal tasks, ensuring effective adaptation to a wide range of legal scenarios while providing accurate and insightful answers.

ChatLaw [54] is a comprehensive legal language model crafted from a deep understanding of the legal domain. It serves as a robust tool for navigating complex legal issues, offering nuanced insights and practical guidance derived from its comprehensive knowledge base and advanced computational techniques. ChatLaw addresses real-world legal challenges by synthesizing legal awareness, relationships, behaviors, and other phenomena within the legal realm. Leveraging prompt engineering within the legal domain, ChatLaw significantly enhances the performance of the Chinese legal language model. During the construction of its training dataset, ChatLaw meticulously fine-tunes and supplements established datasets to ensure relevance and accuracy. It employs RAG technology (Subsection 3.2) to retrieve the latest legal information and utilizes post-retrieval optimization (Subsection 3.2) to further refine its responses and improve overall effectiveness.

ChatDoctor [53] is a pre-trained language model designed to address medical queries and offer advice in medical contexts. One of ChatDoctor's notable features is its external knowledge brain, akin to a Retrieval Augmented Generator (Subsection 3.2). This knowledge repository encompasses a vast array of information on diseases, symptoms, relevant medical tests, and more. Continuously updated and refined, the knowledge brain retrieves new insights from sources such as encyclopedias, medical literature, and other credible references. With its robust dataset and access to a rich knowledge base, ChatDoctor serves as a valuable resource for individuals seeking medical advice. Its ability to understand patient needs and provide tailored recommendations underscores its utility in guiding users toward appropriate medical treatment options.

2) Creative applications

LLMs can be applied in the field of artistic creation, enabling more efficient and intelligent art production.

> **Finding:** With the support of prompt engineering techniques, LLMs are able to generate artistic creations with exceptional efficiency. In both literary and visual arts, *profile* techniques are employed to define specific requirements and details for the creative process. *Knowledge* techniques facilitate the retrieval of multimodal knowledge,enhancing the model's creative capacity. While *reasoning* and *reliability* techniques are less frequently appliedin artistic creation, this is largely due to the inherently creative and less deterministic nature of artistic work.

● Language. CoPoet [56] is a collaborative iterative poetry creation model. Users can iteratively request suggestions from CoPoet through natural language instructions, and CoPoet can generate further content based on user input. Utilizing internal feedback (Subsection 3.3), CoPoet refines the creation process or aids users in their creations by assimilating user feedback.

Ippolito et al. [57] identified barriers to creative writing between AI-driven writing agents and experienced professional writers. They assessed the creative capabilities of the AI writing agent by engaging experts in professional creation from diverse countries, races, and backgrounds. These experts provided open-ended qualitative feedback on the content generated by the AI writing agent. The authors analyze the evaluation results and propose lessons that could enhance the creative abilities of LLMs. Brainstorming emerges as a crucial aspect of creation, and few-shot learning (Subsection 3.1) can be employed to provide prompts for creativity. Task environment information (Subsection 3.1) can help LLMs enhance imagination. By providing environmental information to the language model, detailed data can facilitate more effective brainstorming. Additionally, the authors suggest that this could also be achieved through methods like RAG (Subsection 3.2) and Automatic Post-Editing (APE) [181].

● Vision. As a multi-modal large language model, Sora possesses the capability to process and generate images and videos. Sora excels in a multitude of image and video editing tasks, including seamlessly creating looped videos, animating static images, and extending video duration forwards or backwards in time. Beyond merely inputting text to produce videos, Sora can also utilize pre-existing images or videos as few-shot inputs. This expanded functionality allows Sora to undertake a broader spectrum of editing tasks, such as extending videos and converting images into videos.

● Hearing. ChatMusician [58] adeptly integrates various musical elements, crafting well-structured compositions. By unifying symbolic music understanding and generation tasks, it generates coherent pieces across styles. Utilizing prompt engineering, ChatMusician explores musical creation dynamically. It formats tasks with few-shot prompts (Subsection 3.1) to inspire the creation of LLMs. It also adopts musician role-play prompts (Subsection 3.1) to enhance its perspective.

5 Possible research opportunities

In this section, we outline several promising future research directions that address key challenges in prompt engineering. Although some directions are already covered in existing studies, as discussed in Section 3, we believe these areas warrant further exploration and development.

1) Prompt attack and defense

Due to the unique structure of LLMs, prompts serve as both operational instructions and input data channels. This dual role introduces risks, as specific prompts can be crafted to induce harmful outputs from LLMs, a technique known as prompt attack [154]. Although recent research [182] has proposed defense mechanisms against prompt attacks, few studies focus on prompt engineering as a defense strategy. Adjusting the content or structure of prompts may help prevent generating risky responses. Profile and instruction, in Subsection 3.1, may be useful in such cases, as it provides a framework for organizing prompt content and structure.

2) Thinking powered RAG

Recent studies [183] indicate that applying Retrieval-Augmented Generation (RAG) in all contexts may not consistently enhance LLM capabilities. Integrating RAG with reasoning techniques, such as feedback mechanisms [184], could improve the cognitive efficacy of RAG, facilitating its application in a more contextually appropriate manner. Current methods [184,185] predominantly rely on pre-training and lack interpretability. Developing quantitative approaches to guide retrieval processes and enhance interpretability represents a compelling research direction.

3) Multi-Hop retrieval-augmented generation

The Multi-Hop Question Answering (MHQA) task [23,186] focuses on answering questions that require gathering information from multiple sources and performing multi-step reasoning to arrive at comprehensive answers. Enhancing both the relevance of retrieved documents and the accuracy of reasoning across multiple documents can significantly improve the performance of LLMs on these complex questions. As a straightforward solution, recent studies [31,32] have integrated retrieval into the chain-of-thought reasoning process, enabling LLMs to leverage retrieved documents for answer generation. However, inevitable noise in retrieved documents may mislead LLMs toward incorrect reasoning paths, resulting in erroneous answers [187]. Rather than solely enhancing reasoning capabilities, optimizing retrieved materials through prompt engineering presents a promising approach to improve prompt effectiveness.

4) Advanced reasoning with rationales

Techniques like Chain of Thought (CoT) [30] and Least-to-Most [97] have shown potential for improving LLM reasoning. However, these methods primarily structure reasoning pathways without genuinely enabling LLMs to internalize the underlying rationale. Techniques like CoT establish fixed reasoning pathways for LLMs, akin to rapid brainstorming. Research [188] indicates that slower, more deliberate thinking is often more effective for solving complex problems. While some research [188,189] has explored ways to strengthen the rationale capabilities of LLMs, numerous avenues for further exploration remain. This enhancement in the reasoning process closely aligns with findings in o1 [190], underscoring the substantial potential of these techniques.

5) Domain-specific prompt engineering frameworks

With the expanding application of LLMs across diverse fields, the development of domain-specific prompt engineering frameworks has become a prominent research focus. Optimizing prompts tailored to sectors such as healthcare, law, and finance enables LLMs to produce more accurate, dependable, and efficient outputs [53,169]. As domain-specific agents advance, there is a corresponding rise in specialized agent components and models trained for distinct fields.

Prompt engineering remains central to facilitating interactions between LLMs and various tools [47]. Substantial work is still required to design prompt engineering frameworks that align with the professional standards essential for domain-specific agents.

6) Automatic optimization of prompt

Prompt engineering is a crucial technique directly influencing the responses generated by LLMs. This approach fundamentally differs from programming techniques, as systems based on prompts are guided rather than explicitly programmed. The specific LLM and the details within the prompts both influence the model's output, a topic that will be further discussed in Section 6. Although some research has explored the interpretability of prompts, it typically identifies only the most significant factors that influence the consistency of generated responses. Thus, optimizing prompts remains a challenge. This includes providing explanations for prompts that lead to erroneous answers and automating the optimization of prompts. Exploring automatic optimization within the prompt space and integrating automation with existing prompt techniques represent a practical future research avenue.

6 Summary of findings and insights

● Principles of prompt engineering. Recent studies [33,191] position prompt engineering as an empirical approach. Due to the autoregressive structure of LLMs, prompts function not only as the primary interaction method but also as critical determinants of model outputs. Unlike traditional software engineering, where systems are explicitly programmed, prompt engineering for LLMs relies on guiding model behavior through input modifications. The model's performance is highly sensitive to specific prompt details without any obvious reason those details should matter. Most advanced techniques that leverage this insight, such as the reasoning methods described in Subsection 3.3, are grounded in experimental practices rather than theoretical constructs. Given the impressive capabilities of LLMs, empirically driven approaches to prompt engineering remain essential.

● Prompt design factors. In this paper, we introduce a comprehensive taxonomy for prompt engineering, which synthesizes effective empirical insights into distinct modules for prompt design. The profile and instruction establish the foundational structure of prompts, emphasizing that role information, task information, and example information are critical components in the prompt design. Prompt enhancement techniques are essential for the application of LLMs in real-world scenarios. Currently, knowledge and reasoning are effective strategies for enhancing prompts, as they improve LLM performance by incorporating specific details, such as additional task information and reasoning examples. Although future techniques for augmenting prompt details may emerge, significant research is still required in the domains of knowledge and reasoning. Prior study [192] indicates that the introduction of noise knowledge can enhance LLM performance, highlighting the necessity for more accurate and interpretable knowledge techniques. For instance, employing knowledge graphs can facilitate the acquisition of more precise external knowledge [193]. From an engineering perspective, reliability enhances LLM systems by integrating diverse prompt techniques. It is grounded in the principles of ensemble learning, where techniques such as voting are employed to retain the most stable model responses. Future advancements in prompt engineering can consider these factors comprehensively to design prompts that not only align closely with task requirements but also maximize model performance.

● Post-training impact on prompt engineering. Recently, OpenAI introduced the o1 family of models [190], which have demonstrated breakthrough performance in software engineering and scientific tasks. The o1 model integrates inference processes—such as Chain-of-Thought (CoT) reasoning —into its training framework. By allowing for extended reasoning time, the o1 model enhances its overall problem-solving capabilities. With advancements in such LLMs, reasoning, planning, and reliability techniques are likely to be integrated into future frameworks. At present, profile, instruction, and knowledge technologies remain indispensable due to their relevance to practical tasks.

● Limitations and challenges in prompt engineering. Despite significant advancements in prompt engineering for enhancing large language model (LLM) performance, several critical challenges persist. These challenges pertain to highly ambiguous tasks, adversarial tasks, and scalability in dynamic environments. First, LLMs frequently encounter ambiguous prompts that may lead to multiple plausible answers. Traditional methods often struggle to balance response relevance and diversity. To address this issue, Sun et al. [194] proposed AmbigPrompt —an iterative prompting framework that adaptively guides the LLM to generate distinct and pertinent responses, thereby mitigating ambiguity in user queries. Second, LLMs remain susceptible to adversarial attacks, such as jailbreaking prompts, which can trigger the generation of inappropriate or harmful content. In response, Paulus et al. [195] introduced AdvPrompter, a technique that employs a secondary LLM to rapidly generate human-readable adversarial prompts. This work highlights the urgent need for robust defense mechanisms against such vulnerabilities. Third, scaling prompt engineering techniques to more complex or dynamic environments presents substantial challenges. Kepel and Valogianni [196] addressed this issue with the development of the Automatic Prompt Engineering Toolbox (APET), which enables GPT-4 to autonomously apply prompt engineering techniques. This automation is pivotal for managing scalability concerns effectively. While these advancements offer promising solutions to some of these challenges, further research is necessary to fully overcome these limitations.

7 Conclusion

Prompt engineering is gaining increasing significance across various application domains as a crucial technique for optimizing the performance of large language models (LLMs). Drawing inspiration from the division of agent functions, this survey systematically categorizes prompt engineering techniques into four distinct aspects. This structured approach offers a clear and comprehensive review of the existing research in this area. With a new taxonomy, we examine the applications of LLMs that utilize prompt engineering, presenting specific case studies and empirical results that illustrate the effectiveness of these techniques across different tasks. We aspire for this survey to serve as a comprehensive guide for readers, illuminating the advancements in prompt engineering and offering valuable insights into its practical applications.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2023YFB3002002), the National Natural Science Foundation of China (Grant Nos. 62461146205 and 62322213), Beijing Nova Program (Nos. 20230484397 and 20220484137), the State Key Laboratory of Computer Architecture (ICT, CAS) (CLQ202414), and CCF - ApsaraDB Research Fund (CCF-Aliyun2024003).

Competing interests

The authors declare that they have no competing interests or financial conflicts to disclose.

References

[1] Zhang R, Su Y, Trisedya B D, Zhao X, Yang M, Cheng H, Qi J. AutoAlign: fully automatic and effective knowledge graph alignment enabled by large language models. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(6): 2357−2371

[2] He J, Li Y, Zhai Z, Fang B, Thorne C, Druckenbrodt C, Akhondi S, Verspoor K. Focused contrastive loss for classification with pre-trained language models. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(7): 3047−3061

[3] Zhao Z, Fan W, Li J, Liu Y, Mei X, Wang Y, Wen Z, Wang F, Zhao X, Tang J, Li Q. Recommender systems in the era of large language models (LLMs). IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 6889−6907

[4] Li J, Liu Y, Fan W, Wei X-Y, Liu H, Tang J, Li Q. Empowering molecule discovery for molecule-caption translation with large language models: a ChatGPT perspective. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 6071−6083

[5] Cao Z, Cao C, Xu J, Xu J, Chen Z, Chen Z, Ma X. SCG-tree: shortcut enhanced graph hierarchy tree for efficient spatial queries on massive road networks. Frontiers of Computer Science, 2025, 19(9): 199610

[6] Liu P, Cai P, Zhong K, Li C, Chen H. LRP: learned robust data partitioning for efficient processing of large dynamic queries. Frontiers of Computer Science, 2025, 19(9): 199607

[7] Chen C, Ma W, Gao C, Zhang W, Zeng K, Ye T, Chen Y, Du X. GaussDB-AISQL: a composable cloud-native SQL system with AI capabilities. Frontiers of Computer Science, 2025, 19(9): 199608

[8] Yang J-Q, Dai C, Ou D, Li D, Huang J, Zhan D-C, Zeng X, Yang Y. COURIER: contrastive user intention reconstruction for large-scale visual recommendation. Frontiers of Computer Science, 2025, 19(7): 197602

[9] Zhang T, Zhao J, Yu C, Chen L, Gao Y, Cao B, Fan J, Yu G. Labeling-based centrality approaches for identifying critical edges on temporal graphs. Frontiers of Computer Science, 2025, 19(2): 192601

[10] Wang D, Cai P, Qian W, Zhou A. Efficient and stable quorum-based log replication and replay for modern cluster-databases. Frontiers of Computer Science, 2022, 16(5): 165612

[11] Li X-H, Cao C C, Shi Y, Bai W, Gao H, Qiu L, Wang C, Gao Y, Zhang S, Xue X, Chen L. A survey of data-driven and knowledge-aware explainable AI. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(1): 29−49

[12] Chen L, Zhou X, Yang X, Sellis T. Guest editorial special issue on online recommendation using AI and big data techniques. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(10): 9809−9811

[13] Brown T B, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D M, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D. Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing System. 2020

[14] Zhu Y, Wang Y, Qiang J, Wu X. Prompt-learning for short text classification. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(10): 5328−5339

[15] Xue H, Salim F D. PromptCast: a new prompt-based learning paradigm for time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 6851−6864

[16] Liu J, Fei H, Li F, Li J, Li B, Zhao L, Teng C, Ji D. *TKDP*: threefold knowledge-enriched deep prompt tuning for few-shot named entity recognition. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(11): 6397−6409

[17] Abercrombie G, Curry A C, Dinkar T, Rieser V, Talat Z. Mirages. On anthropomorphism in dialogue systems. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023

[18] Van Buren D. Guided scenarios with simulated expert personae: a remarkable strategy to perform cognitive work. 2023, arXiv preprint arXiv: 2306.03104

[19] Zheng M, Pei J, Logeswaran L, Lee M, Jurgens D. When ”a helpful assistant” is not really helpful: Personas in system prompts do not improve performances of large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024. 2024, 15126–15154

[20] Park J S, O'Brien J, Cai C J, Morris M R, Liang P, Bernstein M S.

Generative agents: interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023
[21] Efrat A, Levy O. The turking test: can language models understand instructions? 2020, arXiv preprint arXiv: 2020
[22] Chen J, Lin H, Han X, Sun L. Benchmarking large language models in retrieval-augmented generation. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence. 2024
[23] Tang Y, Yang Y. MultiHop-RAG: benchmarking retrieval-augmented generation for multi-hop queries. 2024, arXiv preprint arXiv: 2401.15391
[24] Wang L, Yang N, Wei F. Query2doc: query expansion with large language models. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023
[25] Cheng X, Luo D, Chen X, Liu L, Zhao D, Yan R. Lift yourself up: retrieval-augmented text generation with self-memory. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 1899
[26] Jiang Z, Xu F F, Gao L, Sun Z, Liu Q, Dwivedi-Yu J, Yang Y, Callan J, Neubig G. Active retrieval augmented generation. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023. 2023, 7969–7992
[27] Qiao S, Ou Y, Zhang N, Chen X, Yao Y, Deng S, Tan C, Huang F, Chen H. Reasoning with language model prompting: a survey. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023
[28] Huang J, Chang K C C. Towards reasoning in large language models: A survey. In: Findings of the Association for Computational Linguistics: ACL 2023. July 2023, 1049–1065
[29] Yu F, Zhang H, Tiwari P, Wang B. Natural language reasoning, a survey. ACM Computing Surveys, 2024, 56(12): 304
[30] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E D, Le Q V, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing System. 2022, 1800
[31] Long J. Large language model guided tree-of-thought. 2023, arXiv preprint arXiv: 2305.08291
[32] Yao Y, Li Z, Zhao H. Beyond chain-of-thought, effective graph-of-thought reasoning in language models. 2023, arXiv preprint arXiv: 2305.16582
[33] Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. ACM Computing Surveys, 2023, 55(9): 195
[34] Wei J, Tay Y, Bommasani R, Raffel C, Zoph B, Borgeaud S, Yogatama D, Bosma M, Zhou D, Metzler D, Chi E H, Hashimoto T, Vinyals O, Liang P, Dean J, Fedus W. Emergent abilities of large language models. Transactions on Machine Learning Research, 2022. Survey Certification
[35] Sahoo P, Singh A K, Saha S, Jain V, Mondal S, Chadha A. A systematic survey of prompt engineering in large language models: techniques and applications. 2024, arXiv preprint arXiv: 2402.07927
[36] Li H, Leung J, Shen Z. Towards goal-oriented prompt engineering for large language models: a survey. 2024, arXiv preprint arXiv: 2401.14043
[37] Mialon G, Dessì R, Lomeli M, Nalmpantis C, Pasunuru R, Raileanu R, Rozière B, Schick T, Dwivedi-Yu J, Celikyilmaz A, Grave E, LeCun Y, Scialom T. Augmented language models: a survey. Transactions on Machine Learning Research, 2023, 2023
[38] Shin T, Razeghi Y, Logan IV R L, Wallace E, Singh S. AutoPrompt: eliciting knowledge from language models with automatically generated prompts. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. 2020
[39] Petroni F, Rocktäschel T, Riedel S, Lewis P, Bakhtin A, Wu Y, Miller A. Language models as knowledge bases? In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. 2019
[40] Cui L, Wu Y, Liu J, Yang S, Zhang Y. Template-based named entity recognition using BART. In: Proceedings of the Findings of the Association for Computational Linguistics. 2021
[41] Vu T, Lester B, Constant N, Al-Rfou' R, Cer D. SPoT: better frozen model adaptation through soft prompt transfer. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022
[42] Wen Y, Jain N, Kirchenbauer J, Goldblum M, Geiping J, Goldstein T. Hard prompts made easy: gradient-based discrete optimization for prompt tuning and discovery. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 2019
[43] Liu B, Jiang Y, Zhang X, Liu Q, Zhang S, Biswas J, Stone P. LLM+P: empowering large language models with optimal planning proficiency. 2023, arXiv preprint arXiv: 2304.11477
[44] Zhou W, Zhang S, Poon H, Chen M. Context-faithful prompting for large language models. In: Proceedings of the Findings of the Association for Computational Linguistics. 2023
[45] Li S, Ning X, Wang L, Liu T, Shi X, Yan S, Dai G, Yang H, Wang Y. Evaluating quantized large language models. In: Proceedings of the 41st International Conference on Machine Learning. 2024
[46] Dong Q, Li L, Dai D, Zheng C, Ma J, Li R, Xia H, Xu J, Wu Z, Chang B, Sun X, Li L, Sui Z. A survey on in-context learning. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024
[47] Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, Chen Z, Tang J, Chen X, Lin Y, Zhao W X, Wei Z, Wen J. A survey on large language model based autonomous agents. Frontiers of Computer Science, 2024, 18(6): 186345
[48] Yang K, Liu J, Wu J, Yang C, Fung Y R, Li S, Huang Z, Cao X, Wang X, Wang Y, Ji H, Zhai C. If LLM is the wizard, then code is the wand: a survey on how code empowers large language models to serve as intelligent agents. 2024, arXiv preprint arXiv: 2401.00812
[49] Roy D, Zhang X, Bhave R, Bansal C, Las-Casas P, Fonseca R, Rajmohan S. Exploring LLM-based agents for root cause analysis. In: Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering. 2024
[50] Chen M, Tworek J, Jun H, Yuan Q, de Oliveira Pinto H P, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, Ray A, Puri R, Krueger G, Petrov M, Khlaaf H, Sastry G, Mishkin P, Chan B, Gray S, Ryder N, Pavlov M, Power A, Kaiser L, Bavarian M, Winter C, Tillet P, Such F P, Cummings D, Plappert M, Chantzis F, Barnes E, Herbert-Voss A, Guss W H, Nichol A, Paino A, Tezak N, Tang J, Babuschkin I, Balaji S, Jain

S, Saunders W, Hesse C, Carr A N, Leike J, Achiam J, Misra V, Morikawa E, Radford A, Knight M, Brundage M, Murati M, Mayer K, Welinder P, McGrew B, Amodei D, McCandlish S, Sutskever I, Zaremba W. Evaluating large language models trained on code. 2021, arXiv preprint arXiv: 2107.03374

[51] Nijkamp E, Pang B, Hayashi H, Tu L, Wang H, Zhou Y, Savarese S, Xiong C. CodeGen: an open large language model for code with multi-turn program synthesis. In: Proceedings of the 11th International Conference on Learning Representations. 2023

[52] Driess D, Xia F, Sajjadi M S M, Lynch C, Chowdhery A, Ichter B, Wahid A, Tompson J, Vuong Q, Yu T, Huang W, Chebotar Y, Sermanet P, Duckworth D, Levine S, Vanhoucke V, Hausman K, Toussaint M, Greff K, Zeng A, Mordatch I, Florence P. PaLM-E: an embodied multimodal language model. In: Proceedings of the 40th International Conference on Machine Learning. 2023

[53] Li Y, Li Z, Zhang K, Dan R, Jiang S, Zhang Y. ChatDoctor: a medical chat model fine-tuned on a large language model meta-AI (LLaMA) using medical domain knowledge. Cureus, 2023, 15(6): e40895

[54] Yue S, Chen W, Wang S, Li B, Shen C, Liu S, Zhou Y, Xiao Y, Yun S, Huang X, Wei Z. DISC-LawLLM: fine-tuning large language models for intelligent legal services. 2023, arXiv preprint arXiv: 2309.11325

[55] Yiquan W, Yuhang L, Yifei L, Ang L, Siying Z, Kun K. wisdominterrogatory. see https://github.com/zhihaiLLM/wisdomInterrogatory website

[56] Chakrabarty T, Padmakumar V, He H. *Help me write a poem*: instruction tuning as a vehicle for collaborative poetry writing. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022

[57] Ippolito D, Yuan A, Coenen A, Burnam S. Creative writing with an AI-powered writing assistant: perspectives from professional writers. 2022, arXiv preprint arXiv: 2211.05030

[58] Yuan R, Lin H, Wang Y, Tian Z, Wu S, Shen T, Zhang G, Wu Y, Liu C, Zhou Z, Ma Z, Xue L, Wang Z, Liu Q, Zheng T, Li Y, Ma Y, Liang Y, Chi X, Liu R, Wang Z, Lin C, Liu Q, Jiang T, Huang W, Chen W, Chen W, Fu J, Benetos E, Xia G, Dannenberg R, Xue W, Kang S, Guo Y. ChatMusician: understanding and generating music intrinsically with LLM. In: Proceedings of the Findings of the Association for Computational Linguistics. 2024

[59] OpenAI. See openai.com/index/sora/ website, 2024

[60] Reynolds L, McDonell K. Prompt programming for large language models: beyond the few-shot paradigm. In: Proceedings of 2021 CHI Conference on Human Factors in Computing Systems. 2021, 314

[61] Xu X, Tao C, Shen T, Xu C, Xu H, Long G, Lou J-G, Ma S. Re-reading improves reasoning in large language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024

[62] Wang Z, Cai S, Chen G, Liu A, Ma X, Liang Y, Team CraftJarvis. Describe, explain, plan and select: interactive planning with large language models enables open-world multi-task agents. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 1480

[63] Zhu X, Chen Y, Tian H, Tao C, Su W, Yang C, Huang G, Li B, Lu L, Wang X, Qiao Y, Zhang Z, Dai J. Ghost in the minecraft: generally capable agents for open-world environments via large language models with text-based knowledge and memory. 2023, arXiv preprint arXiv: 2305.17144

[64] Kojima T, Gu S S, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 1613

[65] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, others. Language models are unsupervised multitask learners. OpenAI blog, 2019, 1(8): 9

[66] Logan IV R, Balazevic I, Wallace E, Petroni F, Singh S, Riedel S. Cutting down on prompts and parameters: simple few-shot learning with language models. In: Proceedings of the Findings of the Association for Computational Linguistics. 2022

[67] Liu J, Shen D, Zhang Y, Dolan B, Carin L, Chen W. What makes good in-context examples for GPT-3? In: Proceedings of Deep Learning Inside Out (DeeLIO 2022): the 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures. 2022

[68] Su H, Kasai J, Wu C H, Shi W, Wang T, Xin J, Zhang R, Ostendorf M, Zettlemoyer L, Smith N A, Yu T. Selective annotation makes language models better few-shot learners. In: Proceedings of the 11th International Conference on Learning Representations. 2023

[69] Jiang Z, Xu F F, Araki J, Neubig G. How can we know what language models know? Transactions of the Association for Computational Linguistics, 2020, 8: 423−438

[70] Lu Y, Bartolo M, Moore A, Riedel S, Stenetorp P. Fantastically ordered prompts and where to find them: overcoming few-shot prompt order sensitivity. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022

[71] Nie F, Chen M, Zhang Z, Cheng X. Improving few-shot performance of language models via nearest neighbor calibration. 2022, arXiv preprint arXiv: 2212.02216

[72] Fei Y, Hou Y, Chen Z, Bosselut A. Mitigating label biases for in-context learning. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023

[73] Ma H, Zhang C, Bian Y, Liu L, Zhang Z, Zhao P, Zhang S, Fu H, Hu Q, Wu B. Fairness-guided few-shot prompting for large language models. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023

[74] Reif Y, Schwartz R. Beyond performance: quantifying and mitigating label bias in LLMs. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2024

[75] Han Z, Hao Y, Dong L, Sun Y, Wei F. Prototypical calibration for few-shot learning of language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023.

[76] Hu L, Liu Z, Zhao Z, Hou L, Nie L, Li J. A survey of knowledge enhanced pre-trained language models. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(4): 1413−1430

[77] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W T, Rocktäschel T, Riedel S, Kiela D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 793

[78] Pan S, Luo L, Wang Y, Chen C, Wang J, Wu X. Unifying large language models and knowledge graphs: a roadmap. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(7): 3580−3599
[79] Yang L, Chen H, Li Z, Ding X, Wu X. Give us the facts: enhancing large language models with knowledge graphs for fact-aware language modeling. IEEE Transactions on Knowledge and Data Engineering, 2024, 36(7): 3091−3110
[80] Barnett S, Kurniawan S, Thudumu S, Brannelly Z, Abdelrazek M. Seven failure points when engineering a retrieval augmented generation system. In: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI. 2024
[81] Shao Z, Gong Y, Shen Y, Huang M, Duan N, Chen W. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In: Proceedings of the Findings of the Association for Computational Linguistics. 2023
[82] Thakur N, Reimers N, RückléA, Srivastava A, Gurevych I. BEIR: a heterogenous benchmark for zero-shot evaluation of information retrieval models. 2021, arXiv preprint arXiv: 2104.08663
[83] Nguyen T, Rosenberg M, Song X, Gao J, Tiwary S, Majumder R, Deng L. MS MARCO: a human generated machine reading comprehension dataset. In: Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-located with the 30th Annual Conference on Neural Information Processing Systems. 2016
[84] Craswell N, Mitra B, Yilmaz E, Campos D, Voorhees E M. Overview of the TREC 2019 deep learning track. 2020, arXiv preprint arXiv: 2003.07820
[85] Joshi M, Choi E, Weld D, Zettlemoyer L. TriviaQA: a large scale distantly supervised challenge dataset for reading comprehension. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 2017
[86] Ho X, Duong Nguyen A K, Sugawara S, Aizawa A. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In: Proceedings of the 28th International Conference on Computational Linguistics. 2020
[87] Yang Z, Qi P, Zhang S, Bengio Y, Cohen W, Salakhutdinov R, Manning C D. HotpotQA: a dataset for diverse, explainable multi-hop question answering. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. 2018
[88] Yu W, Zhang H, Pan X, Cao P, Ma K, Li J, Wang H, Yu D. Chain-of-note: enhancing robustness in retrieval-augmented language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024
[89] Melz E. Enhancing LLM intelligence with ARM-RAG: auxiliary rationale memory for retrieval augmented generation. 2023, arXiv preprint arXiv: 2311.04177
[90] Cuconasu F, Trappolini G, Siciliano F, Filice S, Campagnano C, Maarek Y, Tonellotto N, Silvestri F. The power of noise: redefining retrieval for RAG systems. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024
[91] Weston J, Sukhbaatar S. System 2 Attention (is something you might need too). 2023, arXiv preprint arXiv: 2311.11829
[92] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017
[93] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G. LLaMA: open and efficient foundation language models. 2023, arXiv preprint arXiv: 2302.13971
[94] Shi F, Chen X, Misra K, Scales N, Dohan D, Chi E, Schärli N, Zhou D. Large language models can be easily distracted by irrelevant context. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 1291
[95] Zhou Y, Liu Z, Jin J, Nie J-Y, Dou Z. Metacognitive retrieval-augmented large language models. In: Proceedings of the ACM Web Conference 2024. 2024
[96] Austin J T, Vancouver J B. Goal constructs in psychology: structure, process, and content. Psychological Bulletin, 1996, 120(3): 338−375
[97] Zhou D, Schärli N, Hou L, Wei J, Scales N, Wang X, Schuurmans D, Cui C, Bousquet O, Le Q V, Chi E H. Least-to-most prompting enables complex reasoning in large language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[98] Zhang Z, Zhang A, Li M, Smola A. Automatic chain of thought prompting in large language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[99] Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. 2019
[100] Diao S, Wang P, Lin Y, Pan R, Liu X, Zhang T. Active prompting with chain-of-thought for large language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. 2024
[101] Lyu Q, Havaldar S, Stein A, Zhang L, Rao D, Wong E, Apidianaki M, Callison-Burch C. Faithful chain-of-thought reasoning. In: Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics. 2023
[102] Liu T, Guo Q, Yang Y, Hu X, Zhang Y, Qiu X, Zhang Z. Plan, verify and switch: integrated reasoning with diverse X-of-thoughts. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023
[103] Liu R, Wei J, Gu S S, Wu T-Y, Vosoughi S, Cui C, Zhou D, Dai A M. Mind’s eye: grounded language model reasoning through simulation. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[104] Li L, Xu J, Dong Q, Zheng C, Sun X, Kong L, Liu Q. Can language models understand physical concepts? In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. 2023
[105] Wang Y, Duan J, Fox D, Srinivasa S. NEWTON: are large language models capable of physical reasoning? In: Proceedings of the Findings of the Association for Computational Linguistics. 2023
[106] Todorov E, Erez T, Tassa Y. MuJoCo: a physics engine for model-based control. In: Proceedings of 2012 IEEE/RSJ International

Conference on Intelligent Robots and Systems. 2012
[107] Drori I, Zhang S, Shuttleworth R, Tang L, Lu A, Ke E, Liu K, Cheng L, Tran S, Cheng N, Wang R, Singh N, Patti T L, Lynch J, Shporer A, Verma N, Wu E, Strang G. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. 2021, arXiv preprint arXiv: 2112.15594
[108] Fu Y, Peng H, Sabharwal A, Clark P, Khot T. Complexity-based prompting for multi-step reasoning. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[109] Cheng Z, Xie T, Shi P, Li C, Nadkarni R, Hu Y, Xiong C, Radev D, Ostendorf M, Zettlemoyer L, Smith N A, Yu T. Binding language models in symbolic languages. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[110] Wu Y, Jiang A Q, Li W, Rabe M N, Staats C, Jamnik M, Szegedy C. Autoformalization with large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 2344
[111] Gao L, Madaan A, Zhou S, Alon U, Liu P, Yang Y, Callan J, Neubig G. PAL: program-aided language models. In: Proceedings of the 40th International Conference on Machine Learning. 2023
[112] Chen W, Ma X, Wang X, Cohen W W. Program of thoughts prompting: disentangling computation from reasoning for numerical reasoning tasks. Transactions on Machine Learning Research, 2023, 2023
[113] Luo X, Zhu Q, Zhang Z, Qin L, Wang X, Yang Q, Xu D, Che W. Multipot: Multilingual program of thoughts harnesses multiple programming languages. arXiv e-prints, 2024, arXiv–2402
[114] Zeng A, Attarian M, Ichter B, Choromanski K M, Wong A, Welker S, Tombari F, Purohit A, Ryoo M S, Sindhwani V, Lee J, Vanhoucke V, Florence P. Socratic models: composing zero-shot multimodal reasoning with language. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[115] Parisi A, Zhao Y, Fiedel N. TALM: tool augmented language models. 2022, arXiv preprint arXiv: 2205.12255
[116] Schick T, Dwivedi-Yu J, Dessí R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T. Toolformer: language models can teach themselves to use tools. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 2997
[117] Taylor R, Kardas M, Cucurull G, Scialom T, Hartshorn A, Saravia E, Poulton A, Kerkez V, Stojnic R. Galactica: a large language model for science. 2022, arXiv preprint arXiv: 2211.09085
[118] Karpas E, Abend O, Belinkov Y, Lenz B, Lieber O, Ratner N, Shoham Y, Bata H, Levine Y, Leyton-Brown K, Muhlgay D, Rozen N, Schwartz E, Shachaf G, Shalev-Shwartz S, Shashua A, Tenenholtz M. MRKL systems: a modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. 2022, arXiv preprint arXiv: 2205.00445
[119] Paranjape B, Lundberg S, Singh S, Hajishirzi H, Zettlemoyer L, Ribeiro M T. ART: automatic multi-step reasoning and tool-use for large language models. 2023, arXiv preprint arXiv: 2303.09014
[120] Lu P, Peng B, Cheng H, Galley M, Chang K-W, Wu Y N, Zhu S-C, Gao J. Chameleon: plug-and-play compositional reasoning with large language models. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023
[121] Patil S G, Zhang T, Wang X, Gonzalez J E. Gorilla: large language model connected with massive APIs. In: Proceedings of the 38th International Conference on Neural Information Processing Systems. 2024
[122] Shen Y, Song K, Tan X, Li D, Lu W, Zhuang Y. HuggingGPT: solving AI tasks with ChatGPT and its friends in hugging face. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023
[123] Tang Q, Deng Z, Lin H, Han X, Liang Q, Cao B, Sun L. ToolAlpaca: generalized tool learning for language models with 3000 simulated cases. 2023, arXiv preprint arXiv: 2306.05301
[124] Ruan J, Chen Y, Zhang B, Xu Z, Bao T, Du G, Shi S, Mao H, Li Z, Zeng X, Zhao R. TPTU: large language model-based AI agents for task planning and tool usage. 2023, arXiv preprint arXiv: 2308.03427
[125] Kong Y, Ruan J, Chen Y, Zhang B, Bao T, Shi S, Qing D, Hu X, Mao H, Li Z, Zeng X, Zhao R, Wang X. TPTU-v2: boosting task planning and tool usage of large language model-based agents in real-world industry systems. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024
[126] Liang Y, Wu C, Song T, Wu W, Xia Y, Liu Y, Ou Y, Lu S, Ji L, Mao S, Wang Y, Shou L, Gong M, Duan N. TaskMatrix.AI: completing tasks by connecting foundation models with millions of APIs. 2023, arXiv preprint arXiv: 2303.16434
[127] Madaan A, Tandon N, Gupta P, Hallinan S, Gao L, Wiegreffe S, Alon U, Dziri N, Prabhumoye S, Yang Y, Gupta S, Majumder B P, Hermann K, Welleck S, Yazdanbakhsh A, Clark P. Self-refine: iterative refinement with self-feedback. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023
[128] Lu J, Zhong W, Huang W, Wang Y, Zhu Q, Mi F, Wang B, Wang W, Zeng X, Shang L, Jiang X, Liu Q. SELF: self-evolution with language feedback. 2023, arXiv preprint arXiv: 2310.00533
[129] Zhang W, Shen Y, Wu L, Peng Q, Wang J, Zhuang Y, Lu W. Self-contrast: better reflection through inconsistent solving perspectives. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. 2024
[130] Wang S, Li S, Sun T, Fu J, Cheng Q, Ye J, Ye J, Qiu X, Huang X. LLM can achieve self-regulation via hyperparameter aware generation. In: Proceedings of the Findings of the Association for Computational Linguistics. 2024
[131] Huang H, Qu Y, Liu J, Yang M, Xu B, Zhao T, Lu W. Self-evaluation of large language model based on glass-box features. In: Proceedings of the Findings of the Association for Computational Linguistics. 2024
[132] Huang J, Chen X, Mishra S, Zheng H S, Yu A W, Song X, Zhou D. Large language models cannot self-correct reasoning yet. In: Proceedings of the 12th International Conference on Learning Representations. 2024
[133] Xu W, Zhu G, Zhao X, Pan L, Li L, Wang W Y. Perils of self-feedback: self-bias amplifies in large language models. 2024, arXiv preprint arXiv: 2402.11436v1
[134] Gou Z, Shao Z, Gong Y, Shen Y, Yang Y, Duan N, Chen W. CRITIC: large language models can self-correct with tool-interactive critiquing. In: Proceedings of the 12th International Conference on Learning Representations. 2024

[135] Zhao Z, Wallace E, Feng S, Klein D, Singh S. Calibrate before use: improving few-shot performance of language models. In: Proceedings of the 38th International Conference on Machine Learning. 2021
[136] Si C, Gan Z, Yang Z, Wang S, Wang J, Boyd-Graber J L, Wang L. Prompting GPT-3 to be reliable. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[137] Liang P, Bommasani R, Lee T, Tsipras D, Soylu D, et al. Holistic evaluation of language models. Transactions on Machine Learning Research, 2023, 2023
[138] Ye X, Durrett G. The unreliability of explanations in few-shot prompting for textual reasoning. 2022, arXiv preprint arXiv: 2205.03401
[139] Shaikh O, Zhang H, Held W, Bernstein M, Yang D. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023
[140] Lin S, Hilton J, Evans O. TruthfulQA: measuring how models mimic human falsehoods. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022
[141] Schick T, Schütze H. Exploiting cloze-questions for few-shot text classification and natural language inference. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics. 2021
[142] Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021
[143] Wang X, Wei J, Schuurmans D, Le Q V, Chi E H, Narang S, Chowdhery A, Zhou D. Self-consistency improves chain of thought reasoning in language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[144] Li Y, Lin Z, Zhang S, Fu Q, Chen B, Lou J-G, Chen W. Making language models better reasoners with step-aware verifier. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023
[145] Pitis S, Zhang M R, Wang A, Ba J. Boosted prompt ensembles for large language models. 2023, arXiv preprint arXiv: 2304.05970
[146] Li Y, Lin Z, Zhang S, Fu Q, Chen B, Lou J-G, Chen W. On the advance of making language models better reasoners. 2022, arXiv preprint arXiv: 2206.02336
[147] Arora S, Narayan A, Chen M F, Orr L J, Guha N, Bhatia K, Chami I, Ré C. Ask me anything: a simple strategy for prompting language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
[148] Hou B, O'Connor J, Andreas J, Chang S, Zhang Y. PromptBoosting: black-box text classification with ten forward passes. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 542
[149] Zhang C, Liu L, Wang C, Sun X, Wang H, Wang J, Cai M. PREFER: prompt ensemble learning via feedback-reflect-refine. In: Proceedings of the 38th AAAI Conference on Artificial Intelligence. 2024
[150] Gao T, Fisch A, Chen D. Making pre-trained language models better few-shot learners. arXiv preprint arXiv:2012.15723, 2020
[151] Allingham J U, Ren J, Dusenberry M W, Gu X, Cui Y, Tran D, Liu J Z, Lakshminarayanan B. A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models. In: Proceedings of the 40th International Conference on Machine Learning. 2023
[152] Meade N, Poole-Dayan E, Reddy S. An empirical survey of the effectiveness of debiasing techniques for pre-trained language models. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics. 2022
[153] Nadeem M, Bethke A, Reddy S. StereoSet: measuring stereotypical bias in pretrained language models. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing. 2021
[154] Liu Y, Jia Y, Geng R, Jia J, Gong N Z. Formalizing and benchmarking prompt injection attacks and defenses. In: Proceedings of the 33rd USENIX Security Symposium. 2024
[155] Tang Z, Zhou K, Li J, Ding Y, Wang P, Yan B, Hua R, Zhang M. CMD: a framework for context-aware model self-detoxification. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. 2024
[156] Kaddour J, Harris J, Mozes M, Bradley H, Raileanu R, McHardy R. Challenges and applications of large language models. 2023, arXiv preprint arXiv: 2307.10169
[157] Bowman R, Cooney O, Newbold J W, Thieme A, Clark L, Doherty G, Cowan B. Exploring how politeness impacts the user experience of chatbots for mental health support. International Journal of Human-Computer Studies, 2024, 184: 103181
[158] Glaese A, McAleese N, Trębacz M, Aslanides J, Firoiu V, Ewalds T, Rauh M, Weidinger L, Chadwick M, Thacker P, Campbell-Gillingham L, Uesato J, Huang P S, Comanescu R, Yang F, See A, Dathathri S, Greig R, Chen C, Fritz D, Elias J S, Green R, Mokrá S, Fernando N, Wu B, Foley R, Young S, Gabriel I, Isaac W, Mellor J, Hassabis D, Kavukcuoglu K, Hendricks L A, Irving G. Improving alignment of dialogue agents via targeted human judgements. 2022, arXiv preprint arXiv: 2209.14375
[159] OpenAI. chatgpt. see the website of OpenAI 2024–5-10
[160] OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, et al. GPT-4 technical report. 2024, arXiv preprint arXiv: 2303.08774
[161] Microsoft. copilot. See the website of copilot.microsoft.com 2024-5-10
[162] Anthropic. The claude 3 model family: opus, sonnet, Haiku. See the website of api.semanticscholar.org/CorpusID:268232499 2024
[163] Gao L, Ma X, Lin J, Callan J. Precise zero-shot dense retrieval without relevance labels. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023
[164] Jagerman R, Zhuang H, Qin Z, Wang X, Bendersky M. Query expansion by prompting large language models. ArXiv, 2023, abs/2305.03653
[165] Jagerman R, Zhuang H, Qin Z, Wang X, Bendersky M. Query expansion by prompting large language models. 2023, arXiv preprint arXiv: 2305.03653
[166] Li M, Zhuang H, Hui K, Qin Z, Lin J, Jagerman R, Wang X, Bendersky M. Generate, filter, and fuse: query expansion via multi-step keyword generation for zero-shot neural rankers. 2023, arXiv preprint

arXiv: 2311.09175
[167] Wu Y, Wu W, Xing C, Zhou M, Li Z. Sequential matching network: a new architecture for multi-turn response selection in retrieval-based chatbots. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. 2017
[168] Ziems N, Yu W, Zhang Z, Jiang M. Large language models are built-in autoregressive search engines. In: Proceedings of the Findings of the Association for Computational Linguistics. 2023
[169] Hongcheng Liu Y M Y WY. L. Xiezhi chinese law large language model. see https://github.com/LiuHC0428/LAW_GPT website, 2023
[170] Singhal K, Azizi S, Tu T, Mahdavi S S, Wei J, et al. Large language models encode clinical knowledge. Nature, 2023, 620(7972): 172−180
[171] Yoo K M, Park D, Kang J, Lee S-W, Park W. GPT3Mix: leveraging large-scale language models for text augmentation. In: Proceedings of the Findings of the Association for Computational Linguistics. 2021, 2225−2239
[172] Bonifacio L, Abonizio H, Fadaee M, Nogueira R. InPars: unsupervised dataset generation for information retrieval. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022
[173] Jeronymo V, Bonifacio L, Abonizio H, Fadaee M, Lotufo D R, Zavrel J, Nogueira R. InPars-v2: large language models as efficient dataset generators for information retrieval. 2023, arXiv preprint arXiv: 2301.01820
[174] Dai H, Liu Z, Liao W, Huang X, Wu Z, Zhao L, Liu W, Liu N, Li S, Zhu D, Cai H, Sun L, Li Q, Shen D, Liu T, Li X. AugGPT: leveraging ChatGPT for text data augmentation. 2023, arXiv preprint arXiv: 2302.13007
[175] Wu S, Irsoy O, Lu S, Dabravolski V, Dredze M, Gehrmann S, Kambadur P, Rosenberg D, Mann G. BloombergGPT: a large language model for finance. 2023, arXiv preprint arXiv: 2303.17564
[176] Yang H, Liu X-Y, Wang C D. FinGPT: open-source financial large language models. 2023, arXiv preprint arXiv: 2306.06031
[177] Luu R K, Buehler M J. BioinspiredLLM: conversational large language model for the mechanics of biological and bio-inspired materials. Advanced Science, 2024, 11(10): 2306724
[178] Liu S, Wang J, Yang Y, Wang C, Liu L, Guo H, Xiao C. ChatGPT-powered conversational drug editing using retrieval and domain feedback. 2023, arXiv preprint arXiv: 2305.18090
[179] Bran A M, Cox S, Schilter O, Baldassari C, White A D, Schwaller P. Augmenting large language models with chemistry tools. Nature Machine Intelligence, 2024, 6(5): 525−535
[180] Zheng Q, Xia X, Zou X, Dong Y, Wang S, Xue Y, Shen L, Wang Z, Wang A, Li Y, Su T, Yang Z, Tang J. CodeGeeX: a pre-trained model for code generation with multilingual benchmarking on HumanEval-X. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2023
[181] Raunak V, Sharaf A, Wang Y, Awadalla H, Menezes A. Leveraging GPT-4 for automatic translation post-editing. In: Proceedings of the Findings of the Association for Computational Linguistics. 2023
[182] Sharma R K, Gupta V, Grossman D. Defending language models against image-based prompt attacks via user-provided specifications. In: Proceedings of 2024 IEEE Security and Privacy Workshops. 2024
[183] Ni S, Bi K, Guo J, Cheng X. When do LLMs need retrieval augmentation? Mitigating LLMs' overconfidence helps retrieval augmentation. In: Proceedings of the Findings of the Association for Computational Linguistics. 2024
[184] Asai A, Wu Z, Wang Y, Sil A, Hajishirzi H. Self-RAG: learning to retrieve, generate, and critique through self-reflection. In: Proceedings of the 12th International Conference on Learning Representations. 2024
[185] Liu Y, Peng X, Zhang X, Liu W, Yin J, Cao J, Du T. RA-ISF: learning to answer and understand from retrieval augmentation via iterative self-feedback. In: Proceedings of the Findings of the Association for Computational Linguistics. 2024
[186] Shi Z, Zhang S, Sun W, Gao S, Ren P, Chen Z, Ren Z. Generate-then-ground in retrieval-augmented generation for multi-hop question answering. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. 2024
[187] Xu S, Pang L, Shen H, Cheng X, Chua T-S. Search-in-the-chain: interactively enhancing large language models with search for knowledge-intensive tasks. In: Proceedings of the ACM Web Conference 2024. 2024
[188] Zelikman E, Wu Y, Mu J, Goodman N D. STaR: bootstrapping reasoning with reasoning. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022
[189] Zelikman E, Harik G, Shao Y, Jayasiri V, Haber N, Goodman N D. Quiet-STaR: language models can teach themselves to think before speaking. 2024, arXiv preprint arXiv: 2403.09629
[190] OpenAI. o1. see https://openai.com/o1/ website2024
[191] Schulhoff S, Ilie M, Balepur N, Kahadze K, Liu A, Si C, Li Y, Gupta A, Han H, Schulhoff S, Dulepet P S, Vidyadhara S, Ki D, Agrawal S, Pham C, Kroiz G C, Li F, Tao H, Srivastava A, Da Costa H, Gupta S, Rogers M L, Goncearenco I, Sarli G, Galynker I, Peskoff D, Carpuat M, White J, Anadkat S, Hoyle A M, Resnik P. The prompt report: a systematic survey of prompt engineering techniques. 2024, arXiv preprint arXiv: 2406.06608
[192] Yoran O, Wolfson T, Ram O, Berant J. Making retrieval-augmented language models robust to irrelevant context. arXiv preprint arXiv:2310.01558, 2023
[193] Microsoft. GraphRAG. See the website of www.microsoft.com/en-us/research/project/graphrag/, 2024
[194] Sun W, Cai H, Chen H, Ren P, Chen Z, de Rijke M, Ren Z. Answering ambiguous questions via iterative prompting. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. 2023, 7669−7683
[195] Paulus A, Zharmagambetov A, Guo C, Amos B, Tian Y. AdvPrompter: fast adaptive adversarial prompting for LLMs. 2024, arXiv preprint arXiv: 2404.16873
[196] Kepel D, Valogianni K. Autonomous prompt engineering in large language models. 2024, arXiv preprint arXiv: 2407.11000

Yao-Yang LIU joined the Key Laboratory of Data Engineering and Knowledge Engineer (MOE), Renmin University of China, China in 2023. His major research interests include machine learning, database, and large language models.

Zhen ZHENG received his PhD degree from the Department of Computer Science and Technology, Tsinghua University, China in 2019. He joined Alibaba in August 2019. His research interests include AI compiler, large scale machine learning systems, and heterogeneous computing.

Feng ZHANG received his PhD degree in computer science from Tsinghua University, China in 2017. He is a professor with the Key Laboratory of Data Engineering and Knowledge Engineer (MOE), Renmin University of China, China. His major research interests include big data management systems and parallel and distributed systems.

Jin-Cheng FENG received her bachelor degree from the School of Information, Renmin University of China, China in 2025. She will begin her study for the master’s degree at the Gaoling School of Artificial Intelligence, Renmin University of China, China from September 2025.

Yi-Yang FU joined Knowledge reasoning and cognition, Renmin University of China, China in 2023. She is now a junior student at School of Information, Renmin University of China, China.

Ji-Dong ZHAI received his BS degree in computer science from the University of Electronic Science and Technology of China, China in 2003, and PhD degree in computer science from Tsinghua University, China in 2010. He is an associate professor in Department of Computer Science and Technology, Tsinghua University, China. His research interests include performance evaluation for high performance computers, performance analysis, and modeling of parallel applications.

Bing-Sheng HE received his bachelor degree in computer science from Shanghai Jiao Tong University, China (1999−2003), and his PhD degree in computer science from Hong Kong University of Science and Technology, China (2003−2008). He is an associate professor in School of Computing, National University of Singapore, Singapore. His research interests are high performance computing, distributed and parallel systems, and database systems.

Xiao ZHANG received his master’s degree from Renmin University and his PhD degree from the Institute of Computing Technology, Chinese Academy of Science, China in 1998 and 2001, respectively, both in computer science and technology. He is a professor at the School of Information, Renmin University of China, China. His research interests fouces on big data management systems.

Xiao-Yong DU obtained his BS degree from Hangzhou University, China in 1983, his ME degree from Renmin University of China, China in 1988, and his PhD degree from Nagoya Institute of Technology, Japan in 1997. He is currently a professor with the School of Information, Renmin University of China, China. His current research interests include databases and intelligent information retrieval.

A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert,
Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt
Department of Computer Science
Vanderbilt University, Tennessee
Nashville, TN, USA
{jules.white, quchen.fu, george.s.hays, michael.sandborn, carlos.olea, henry.gilbert,
ashraf.elnashar, jesse.spencer-smith, douglas.c.schmidt}@vanderbilt.edu

***Abstract*—Prompt engineering is an increasingly important skill set needed to converse effectively with large language models (LLMs), such as ChatGPT. Prompts are instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output. Prompts are also a form of programming that can customize the outputs and interactions with an LLM.**

This paper describes a catalog of prompt engineering techniques presented in pattern form that have been applied to solve common problems when conversing with LLMs. Prompt patterns are a knowledge transfer method analogous to software patterns since they provide reusable solutions to common problems faced in a particular context, i.e., output generation and interaction when working with LLMs.

This paper provides the following contributions to research on prompt engineering that apply LLMs to automate software development tasks. First, it provides a framework for documenting patterns for structuring prompts to solve a range of problems so that they can be adapted to different domains. Second, it presents a catalog of patterns that have been applied successfully to improve the outputs of LLM conversations. Third, it explains how prompts can be built from multiple patterns and illustrates prompt patterns that benefit from combination with other prompt patterns.

***Index Terms*—large language models, prompt patterns, prompt engineering**

I. Introduction

Conversational large language models (LLMs) [1], such as ChatGPT [2], have generated immense interest in a range of domains for tasks ranging from answering questions on medical licensing exams [3] to generating code snippets. This paper focuses on enhancing the application of LLMs in several domains, such as helping developers code effectively and efficiently with unfamiliar APIs or allowing students to acquire new coding skills and techniques.

LLMs are particularly promising in domains where humans and AI tools work together as trustworthy collaborators to more rapidly and reliably evolve software-reliant systems [4]. For example, LLMs are being integrated directly into software tools, such as Github's Co-Pilot [5]–[7] and included in integrated development environments (IDEs), such as IntelliJ [8] and Visual Studio Code, thereby allowing software teams to access these tools directly from their preferred IDE.

A prompt [9] is a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing or refining its capabilities. A prompt can influence subsequent interactions with—and output generated from—an LLM by providing specific rules and guidelines for an LLM conversation with a set of initial rules. In particular, a prompt sets the context for the conversation and tells the LLM what information is important and what the desired output form and content should be.

For example, a prompt could specify that an LLM should only generate code that follows a certain coding style or programming paradigm. Likewise, it could specify that an LLM should flag certain keywords or phrases in a generated document and provide additional information related to those keywords. By introducing these guidelines, prompts facilitate more structured and nuanced outputs to aid a large variety of software engineering tasks in the context of LLMs.

Prompt engineering is the means by which LLMs are programmed via prompts. To demonstrate the power of prompt engineering, we provide the following prompt:

> **Prompt:** "From now on, I would like you to ask me questions to deploy a Python application to AWS. When you have enough information to deploy the application, create a Python script to automate the deployment."

This example prompt causes ChatGPT to begin asking the user questions about their software application. ChatGPT will drive the question-asking process until it reaches a point where it has sufficient information to generate a Python script that automates deployment. This example demonstrates the programming potential of prompts beyond conventional "generate a method that does X" style prompts or "answer this quiz question".

Moreover, prompts can be engineered to program an LLM to accomplish much more than simply dictating the output type or filtering the information provided to the model. With the right prompt, it is possible to create entirely new interaction paradigms, such as having an LLM generate and give a quiz associated with a software engineering concept or tool, or even simulate a Linux terminal window. Moreover, prompts have the potential for self-adaptation, suggesting other prompts to gather additional information or generate related artifacts. These advanced capabilities of prompts highlight the importance of engineering them to provide value beyond simple text or code generation.

Prompt patterns are essential to effective prompt engineering. A key contribution of this paper is the introduction of *prompt patterns* to document successful approaches for systematically engineering different output and interaction goals when working with conversational LLMs. We focus largely on engineering domain-independent prompt patterns and introduce a catalog of essential prompt patterns to solve problems ranging from production of visualizations and code artifacts to automation of output steps that help fact check outputs.

The remainder of this paper is organized as follows: Section II introduces prompt patterns and compares these patterns to well-known software patterns [10]; Section III describes 16 prompt patterns that have been applied to solve common problems in the domain of conversational LLM interaction and output generation for automating software development tasks; Section IV discusses related work; and Section V presents concluding remarks and lessons learned.

II. Comparing Software Patterns with Prompt Patterns

The quality of the output(s) generated by a conversational LLM is directly related to the quality of the prompts provided by the user. As discussed in Section I, the prompts given to a conversational LLM can be used to program interactions between a user and an LLM to better solve a variety of problems. One contribution of this paper is the framework it provides to document patterns that structure prompts to solve a range of software tasks that can be adapted to different domains.

This framework is useful since it focuses on codifying patterns that can be applied to help users better interact with conversational LLMs in a variety of contexts, rather than simply discussing interesting examples or domain-specific prompts. Codifying this knowledge in pattern form enhances reuse and transferability to other contexts and domains where users face similar—but not identical—problems.

The topic of knowledge transfer has been studied extensively in the software patterns literature [10], [11] at multiple levels, *e.g.*, design, architectural, and analysis. This paper applies a variant of a familiar pattern form as the basis of our prompt engineering approach. Since prompts are a form of programming, it is natural to document them in pattern form.

A. Overview of Software Patterns

A software pattern provides a reusable solution to a recurring problem within a particular context [10]. Documenting software patterns concisely conveys (and generalizes) from specific problems being addressed to identify important forces and/or requirements that should be resolved and/or addressed in successful solutions.

A pattern form also includes guidance on how to implement the pattern, as well as information on the trade-offs and considerations to take into account when implementing a pattern. Moreover, example applications of the pattern are often provided to further showcase the pattern's utility in practice. Software patterns are typically documented in a stylized form to facilitate their use and understanding, such as:

- **A name and classification**. Each pattern has a name that identifies the pattern and should be used consistently. A classification groups patterns into broad categories, such as creational, structural, or behavioral.
- **The intent** concisely conveys the purpose the pattern is intended to achieve.
- **The motivation** documents the underlying problem the pattern is meant to solve and the importance of the problem.
- **The structure and participants**. The structure describes the different pattern participants (such as classes and objects) and how they collaborate to form a generalized solution.
- **Example code** concretely maps the pattern to some underlying programming language(s) and aids developers in gaining greater insight into how that pattern can be applied effectively.
- **Consequences** summarize the pros and cons of applying the pattern in practice.

B. Overview of Prompt Patterns

Prompt patterns are similar to software patterns in that they offer reusable solutions to specific problems. They focus more specifically, however, on the context of output generation from large-scale language models (LLMs), such as ChatGPT. Just as software patterns provide a codified approach to solving common software development challenges, prompt patterns provide a codified approach to customizing the output and interactions of LLMs.

By documenting and leveraging prompt patterns in the context of automating software development tasks, individual users and teams can enforce constraints on the generated output, ensure that relevant information is included, and change the format of interaction with the LLM to better solve problems they face. Prompt patterns can be viewed as a corollary to the broad corpus of general software patterns, just adapted to the more specific context of LLM output generation.

Prompt patterns follow a similar format to classic software patterns, with slight modifications to match the context of output generation with LLMs.[1] Each of the analogous sections for the prompt pattern form used in this paper is summarized below:

- **A name and classification**. The prompt pattern name uniquely identifies the pattern and ideally indicates the problem that is being addressed. For the classification, we have developed a series of initial categories of pattern types, which are summarized in Table I and include **Output Customization**, **Error Identification**, **Prompt Improvement**, **Interaction**, and **Context Control**.
- **The intent and context** describes the problem the prompt pattern solves and the goals it achieves. The problem

[1] The most direct translation of software pattern structure to prompt patterns is the naming, intent, motivation, and sample code. The structure and classification, however, although named similarly, require more adaptation.

should ideally be independent of any domain, though domain-specific patterns may also be documented with an appropriate discussion of the context where the pattern applies.

- **The motivation** provides the rationale for the problem and explains why solving it is important. The motivation is explained in the context of users interacting with a conversational LLM and how it can improve upon users informally prompting the LLM in one or more circumstances. Specific circumstances where the improvements are expected are documented.
- **The structure and key ideas**. The structure describes the fundamental contextual information, as a series of key ideas, that the prompt pattern provides to the LLM. These ideas are similar to "participants" in a software pattern. The contextual information may be communicated through varying wording (just as a software pattern can have variations in how it is realized in code), but should have fundamental pieces of information that form a core element of the pattern.
- **Example implementation** demonstrates how the prompt pattern is worded in practice.
- **Consequences** summarize the pros and cons of applying the pattern and may provide guidance on how to adapt the prompt to different contexts.

C. Evaluating Means for Defining a Prompt Pattern's Structure and Ideas

In software patterns, the structure and participants are normally defined in terms of UML diagrams, such as structure diagrams and/or interaction diagrams. These UML diagrams explain what the participants of the pattern are and how they interact to solve the problem. In prompt patterns, something analogous is needed, though UML may not be an appropriate structural documentation approach since it is intended to describe software structures, as opposed to the ideas to communicate in a prompt.

Several possible approaches could be used, ranging from diagrams to defining grammars for a prompt language. Although grammars may seem attractive due to their formal nature, they also incur the following challenges:

- The goal of prompts is to communicate knowledge in a clear and concise way to conversation LLM users, who may or may not be computer scientists or programmers. As a community, we should strive to create an approachable format that communicates knowledge clearly to a diverse target audience.
- It is possible to phrase a prompt in many different ways. It is hard, however, to define a grammar that accurately and completely expresses all the nuanced ways that components of a prompt could be expressed in text or symbols.
- Prompts fundamentally convey ideas to a conversational LLM and are not simply the production of tokens for input. In particular, an idea built into a prompt pattern can be communicated in many ways and its expression should be at a higher-level than the underlying tokens representing the idea.
- It is possible to program an LLM to introduce novel semantics for statements and words that create new ways for communicating an idea. In contrast, grammars may not easily represent ideas that can be expressed through completely new symbology or languages that the grammar designer was not aware of.

D. A Way Forward: Fundamental Contextual Statements

An open research question, therefore, is what approach is more effective than formal grammars for describing prompt pattern structure and ideas. We propose the concept of *fundamental contextual statements*, which are written descriptions of the important ideas to communicate in a prompt to an LLM. An idea can be rewritten and expressed in arbitrary ways based on user needs and experience. The key ideas to communicate, however, are presented to the user as a series of simple, but fundamental, statements.

One benefit of adopting and applying the fundamental contextual statements approach is that it is intentionally intuitive to users. In particular, we expect users will understand how to express and adapt the statements in a contextually appropriate way for their domain. Moreover, since the underlying ideas of the prompt are captured, these same ideas can be expressed by the user in alternate symbology or wording that has been introduced to the LLM using patterns, such as the *Meta Language Creation* pattern presented in Section III-B.

Our ultimate goal is to enhance prompt engineering by providing a framework for designing prompts that can be reused and/or adapted to other LLMs in the same way that software patterns can be implemented in different programming languages and platforms. For the purposes of this paper, however, all prompts were tested with ChatGPT [12] using the ChatGPT+ service. We use ChatGPT as the LLM for all examples presented in this paper due to its widespread availability and popularity. These examples were documented through a combination of exploring the corpus of community-posted prompts on the Internet and independent prompt creation from our use of ChatGPT to automating software development tasks.

III. A Catalog of Prompt Patterns for Conversational LLMs

This section presents our catalog of prompt patterns that have been applied to solve common problems in the domain of conversational LLM interaction and output generation for automating software tasks. Each prompt pattern is accompanied by concrete implementation samples and examples with and without the prompt.

A. Summary of the Prompt Pattern Catalog

The classification of prompt patterns is an important consideration in documenting the patterns. Table I outlines the initial classifications for the catalog of prompt patterns we identified in our work with ChatGPT thus far.

TABLE I
CLASSIFYING PROMPT PATTERNS

Pattern Category	Prompt Pattern
Input Semantics	*Meta Language Creation*
Output Customization	*Output Automater* *Persona* *Visualization Generator* *Recipe* *Template*
Error Identification	*Fact Check List* *Reflection*
Prompt Improvement	*Question Refinement* *Alternative Approaches* *Cognitive Verifier* *Refusal Breaker*
Interaction	*Flipped Interaction* *Game Play* *Infinite Generation*
Context Control	*Context Manager*

As shown in this table, there are five categories of prompt patterns in our classification framework: **Input Semantics**, **Output Customization**, **Error Identification**, **Prompt Improvement**, and **Interaction**, each of which is summarized below.

The **Input Semantics** category deals with how an LLM understands the input and how it translates the input into something it can use to generate output. This category includes the *Meta Language Creation* pattern, which focuses on creating a custom language for the LLM to understand. This pattern is useful when the default input language is ill-suited for expressing ideas the user wants to convey to the LLM.

The **Output Customization** category focuses on constraining or tailoring the types, formats, structure, or other properties of the output generated by the LLM. The prompt patterns in this category include *Output Automater*, *Persona*, *Visualization Generator*, *Recipe*, and *Template* patterns. The *Output Automater* pattern allows the user to create scripts that can automate any tasks the LLM output suggests the user should perform. The *Persona* pattern gives the LLM a persona or role to play when generating output. The *Visualization Generator* pattern allows the user to generate visualizations by producing textual outputs that can be fed to other tools, such as other AI-based image generators, like DALL-E [13]. The *Recipe* pattern allows the user to obtain a sequence of steps or actions to realize a stated end result, possibly with partially known information or constraints. The *Template* pattern allows the user to specify a template for the output, which the LLM fills in with content.

The **Error Identification** category focuses on identifying and resolving errors in the output generated by the LLM. This category includes the *Fact Check List* and *Reflection* patterns. The *Fact Check List* pattern requires the LLM to generate a list of facts the output depends on that should be fact-checked. The *Reflection* pattern requires the LLM to introspect on its output and identify any errors.

The **Prompt Improvement** category focuses on improving the quality of the input and output. This category includes the *Question Refinement*, *Alternative Approaches*, *Cognitive Verifier*, and *Refusal Breaker* patterns. The *Question Refinement* pattern ensures the LLM always suggests a better version of the user's question. The *Alternative Approaches* pattern requires the LLM to suggest alternative ways of accomplishing a user-specified task. The *Cognitive Verifier* pattern instructs the LLM to automatically suggest a series of subquestions for the user to answer before combining the answers to the subquestions and producing an answer to the overall question. The *Refusal Breaker* pattern requires the LLM to automatically reword the user's question when it refuses to produce an answer.

The **Interaction category** focuses on the interaction between the user and the LLM. This category includes the *Flipped Interaction*, *Game Play*, and *Infinite Generation* patterns. The *Flipped Interaction* pattern requires the LLM to ask questions rather than generate output. The *Game Play* pattern requires the LLM to generate output in the form of a game. The *Infinite Generation* pattern requires the LLM to generate output indefinitely without the user having to reenter the generator prompt each time.

Finally, the **Context Control** category focuses on controlling the contextual information in which the LLM operates. This category includes the *Context Manager* pattern, which allows the user to specify the context for the LLM's output.

The remainder of this section describes each of these prompt patterns using the pattern form discussed in Section II-B.

B. The Meta Language Creation Pattern

1) Intent and Context: During a conversation with an LLM, the user would like to create the prompt via an alternate language, such as a textual short-hand notation for graphs, a description of states and state transitions for a state machine, a set of commands for prompt automation, etc. The intent of this pattern is to explain the semantics of this alternative language to the LLM so the user can write future prompts using this new language and its semantics.

2) Motivation: Many problems, structures, or other ideas communicated in a prompt may be more concisely, unambiguously, or clearly expressed in a language other than English (or whatever conventional human language is used to interact with an LLM). To produce output based on an alternative language, however, an LLM needs to understand the language's semantics.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
When I say X, I mean Y (or would like you to do Y)

The key structure of this pattern involves explaining the meaning of one or more symbols, words, or statements to the LLM so it uses the provided semantics for the ensuing conversation. This description can take the form of a simple translation, such as "X" means "Y". The description can also take more complex forms that define a series of commands and their semantics, such as "when I say X, I want you to do ". In this case, "X" is henceforth bound to the semantics of "take action".

4) Example Implementation: The key to successfully using the *Meta Language Creation* pattern is developing an unambiguous notation or shorthand, such as the following:

> "From now on, whenever I type two identifiers separated by a "→", I am describing a graph. For example, "a → b" is describing a graph with nodes "a" and "b" and an edge between them. If I separate identifiers by "-[w:2, z:3]→", I am adding properties of the edge, such as a weight or label."

This example of the *Meta Language Creation* pattern establishes a standardized notation for describing graphs by defining a convention for representing nodes and edges. Whenever the author types two identifiers separated by a "→" symbol, it is an indication that a graph is being described. For example, if the author types "a → b", this indicates that a graph is being defined with nodes "a" and "b", and that there is an edge between them. This convention provides a clear and concise way to communicate the structure of a graph in written form.

Moreover, the prompt goes on to specify that additional information about the edges, such as a weight or label, can be provided using the syntax "-[w:2, z:3]→". This notation allows for the specification of additional properties beyond the basic structure of the graph. The specified properties are associated with the edge between the two nodes and can provide important context for the interpretation of the graph. This standardized notation for describing graphs can make it easier to communicate graph structures and properties, which may not be easy or are very verbose to describe as a series of sentences.

5) Consequences: Although this pattern provides a powerful means to customize a user's interaction with an LLM, it may create the potential for confusion within the LLM. As important as it is to clearly define the semantics of the language, it is also essential to ensure the language itself introduces no ambiguities that degrade the LLM's performance or accuracy. For example, the prompt "whenever I separate two things by commas, it means that the first thing precedes the second thing" will likely create significant potential for ambiguity and create the potential for unexpected semantics if punctuation involving commas is used in the prompt.

To showcase the potential for confusion, ChatGPT will warn the user and potentially be unable to perform a specific mapping of a symbol or term to a new meaning. For example, if following prompt is given to ChatGPT

> "Whenever I say "a", I am referring to Marie Antoinette."

it will respond that this prompt creates too much confusion as "a" is an indefinite article and too commonly used, as shown below:

> "Referring to Marie Antoinette using "a" could potentially cause confusion, as "a" is an indefinite article commonly used to refer to a noun that has not been mentioned before. It may be more clear to use her name, "Marie Antoinette", or a pronoun such as "she" or "her" to refer to her. Moreover, using context to establish who you are referring to can also help avoid confusion in conversation."

In general, an LLM will perform better on data it was trained on. Therefore, when an existing notation is available—but may not be the dominant meaning—the *Meta Language Creation* pattern can provide context to scope the meaning of the symbol to improve the accurately and utility of the output. For example, the symbol "→" conveys slightly different information in graphs and propositional logic. The statement "p → q, q → p" may therefore make sense, but only if the LLM is given the appropriate context through a *Meta Language Creation* prompt.

Although *Meta Language Creation* is a powerful pattern it must be used carefully, particularly when describing concepts that may otherwise be hard to precisely or concisely describe. These types of prompts are thus best used in completely new conversation sessions. Using a single meta-language-per-conversation session may also be a best practice since it avoids the potential for conflicting or unexpected semantics being applied to the conversation over time.

C. The Output Automater Pattern

1) Intent and Context: The intent of this pattern is to have the LLM generate a script or other automation artifact that can automatically perform any steps it recommends taking as part of its output. The goal is to reduce the manual effort needed to implement any LLM output recommendations.

2) Motivation: The output of an LLM is often a sequence of steps for the user to follow. For example, when asking an LLM to generate a Python configuration script it may suggest a number of files to modify and changes to apply to each file. However, having users continually perform the manual steps dictated by LLM output is tedious and error-prone.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Whenever you produce an output that has at least one step to take and the following properties (alternatively, always do this)
Produce an executable artifact of type X that will automate these steps

The first part of the pattern identifies the situations under which automation should be generated. A simple approach is to state that the output includes at least two steps to take and that an automation artifact should be produced. The scoping is up to the user, but helps prevent producing an output automation scripts in cases where running the output automation script will take more user effort than performing the original steps produced in the output. The scope can be limited to outputs requiring more than a certain number of steps.

The next part of this pattern provides a concrete statement of the type of output the LLM should output to perform the automation. For example, "produce a Python script" gives the LLM a concrete understanding to translate the general steps into equivalent steps in Python. The automation artifact should be concrete and must be something that the LLM associates with the action of "automating a sequence of steps".

4) Example Implementation: A sample of this prompt pattern applied to code snippets generated by the ChatGPT LLM is shown below:

> "From now on, whenever you generate code that spans more than one file, generate a Python script that can be run to automatically create the specified files or make changes to existing files to insert the generated code."

This pattern is particularly effective in software engineering as a common task for software engineers using LLMs is to then copy/paste the outputs into multiple files. Some tools, such as Copilot, insert limited snippets directly into the section of code that the coder is working with, but tools, such as ChatGPT, do not provide these facilities. This automation trick is also effective at creating scripts for running commands on a terminal, automating cloud operations, or reorganizing files on a file system.

This pattern is a powerful complement for any system that can be computer controlled. The LLM can provide a set of steps that should be taken on the computer-controlled system and then the output can be translated into a script that allows the computer controlling the system to automatically take the steps. This is a direct pathway to allowing LLMs, such as ChatGPT, to integrate quality into—and to control—new computing systems that have a known scripting interface.

5) Consequences: An important usage consideration of this pattern is that the automation artifact must be defined concretely. Without a concrete meaning for how to "automate" the steps, the LLM often states that it "can't automate things" since that is beyond its capabilities. LLMs typically accept requests to produce code, however, so the goal is to instruct the LLM to generate text/code, which can be executed to automate something. This subtle distinction in meaning is important to help an LLM disambiguate the prompt meaning.

One caveat of the *Output Automater* pattern is the LLM needs sufficient conversational context to generate an automation artifact that is functional in the target context, such as the file system of a project on a Mac vs. Windows computer. This pattern works best when the full context needed for the automation is contained within the conversation, *e.g.*, when a software application is generated from scratch using the conversation and all actions on the local file system are performed using a sequence of generated automation artifacts rather than manual actions unknown to the LLM. Alternatively, self-contained sequences of steps work well, such as "how do I find the list of open ports on my Mac computer".

In some cases, the LLM may produce a long output with multiple steps and not include an automation artifact. This omission may arise for various reasons, including exceeding the output length limitation the LLM supports. A simple workaround for this situation is to remind the LLM via a follow-on prompt, such as "But you didn't automate it", which provides the context that the automation artifact was omitted and should be generated.

At this point in the evolution of LLMs, the *Output Automater* pattern is best employed by users who can read and understand the generated automation artifact. LLMs can (and do) produce inaccuracies in their output, so blindly accepting and executing an automation artifact carries significant risk. Although this pattern may alleviate the user from performing certain manual steps, it does not alleviate their responsibility to understand the actions they undertake using the output. When users execute automation scripts, therefore they assume responsibility for the outcomes.

D. The Flipped Interaction Pattern

1) Intent and Context: You want the LLM to ask questions to obtain the information it needs to perform some tasks. Rather than the user driving the conversation, therefore, you want the LLM to drive the conversation to focus it on achieving a specific goal. For example, you may want the LLM to give you a quick quiz or automatically ask questions until it has sufficient information to generate a deployment script for your application to a particular cloud environment.

2) Motivation: Rather than having the user drives a conversation, an LLM often has knowledge it can use to more accurately obtain information from the user. The goal of the *Flipped Interaction* pattern is to flip the interaction flow so the LLM asks the user questions to achieve some desired goal. The LLM can often better select the format, number, and content of the interactions to ensure that the goal is reached faster, more accurately, and/or by using knowledge the user may not (initially) possess.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
I would like you to ask me questions to achieve X
You should ask questions until this condition is met or to achieve this goal (alternatively, forever)
(Optional) ask me the questions one at a time, two at a time, etc.

A prompt for a flipped interaction should always specify the goal of the interaction. The first idea (*i.e.*, you want the LLM to ask questions to achieve a goal) communicates this goal to the LLM. Equally important is that the questions should focus on a particular topic or outcome. By providing the goal, the LLM can understand what it is trying to accomplish through the interaction and tailor its questions accordingly. This "inversion of control" enables more focused and efficient interaction since the LLM will only ask questions that it deems relevant to achieving the specified goal.

The second idea provides the context for how long the interaction should occur. A flipped interaction can be terminated with a response like "stop asking questions". It is often better, however, to scope the interaction to a reasonable length or only as far as is needed to reach the goal. This goal can be surprisingly open-ended and the LLM will continue to work towards the goal by asking questions, as is the case in the example of "until you have enough information to generate a Python script".

By default, the LLM is likely to generate multiple questions per iteration. The third idea is completely optional, but can improve usability by limiting (or expanding) the number of questions that the LLM generates per cycle. If a precise number/format for the questioning is not specified, the questioning will be semi-random and may lead to one-at-a-time questions or ten-at-a-time questions. The prompt can thus be tailored to include the number of questions asked at a time, the order of the questions, and any other formatting/ordering considerations to facilitate user interaction.

4) Example Implementation: A sample prompt for a flipped interaction is shown below:

> "From now on, I would like you to ask me questions to deploy a Python application to AWS. When you have enough information to deploy the application, create a Python script to automate the deployment."

In general, the more specific the prompt regarding the constraints and information to collect, the better the outcome. For instance, the example prompt above could provide a menu of possible AWS services (such as Lambda, EC2, etc.) with which to deploy the application. In other cases, the LLM may be permitted to simply make appropriate choices on its own for things that the user doesn't explicitly make decisions about. One limitation of this prompt is that, once other contextual information is provided regarding the task, it may require experimentation with the precise phrasing to get the LLM to ask the questions in the appropriate number and flow to best suit the task, such as asking multiple questions at once versus one question at a time.

5) Consequences: One consideration when designing the prompt is how much to dictate to the LLM regarding what information to collect prior to termination. In the example above, the flipped interaction is open-ended and can vary significantly in the final generated artifact. This open-endedness makes the prompt generic and reusable, but may potentially ask additional questions that could be skipped if more context is given.

If specific requirements are known in advance, it is better to inject them into the prompt rather than hoping the LLM will obtain the needed information. Otherwise, the LLM will non-nondeterministically decide whether to prompt the user for the information or make an educated guess as to an appropriate value.

For example, the user can state that they would like to deploy an application to Amazon AWS EC2, rather than simply state "the cloud" and require multiple interactions to narrow down the deployment target. The more precise the initial information, the better the LLM can use the limited questions that a user is likely willing to answer to obtain information to improve its output.

When developing prompts for flipped interactions, it is important to consider the level of user knowledge, engagement, and control. If the goal is to accomplish the goal with as little user interaction as possible (minimal control), that should be stated explicitly.Conversely, if the goal is to ensure the user is aware of all key decisions and confirms them (maximum engagement) that should also be stated explicitly. Likewise, if the user is expected to have minimal knowledge and should have the questions targeted at their level of expertise, this information should be engineered into the prompt.

E. The Persona Pattern

1) Intent and Context: In many cases, users would like LLM output to always take a certain point of view or perspective. For example, it may be useful for to conduct a code review as if the LLM was a security expert. The intent of this pattern is to give the LLM a "persona" that helps it select what types of output to generate and what details to focus on.

2) Motivation: Users may not know what types of outputs or details are important for an LLM to focus on to achieve a given task. They may know, however, the role or type of person that they would normally ask to get help with these things. The *Persona* pattern enables the users to express what they need help with without knowing the exact details of the outputs they need.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Act as persona X
Provide outputs that persona X would create

The first statement conveys the idea that the LLM needs to act as a specific persona and provide outputs that such a persona would. This persona can be expressed in a number of ways, ranging from a job description, title, fictional character, historical figure, etc. The persona should elicit a set of attributes associated with a well-known job title, type of person, etc.[2]

The secondary idea—provide outputs that persona X would create—offers opportunities for customization. For example, a teacher might provide a large variety of different output types, ranging from assignments to reading lists to lectures. If a more specific scope to the type of output is known, the user can provide it in this statement.

[2]Be aware, however, that personas relating to living people or people considered harmful make be disregarded due to underlying LLM privacy and security rules.

4) Example Implementation: A sample implementation for code review is shown below:

> "From now on, act as a security reviewer. Pay close attention to the security details of any code that we look at. Provide outputs that a security reviewer would regarding the code."

In this example, the LLM is instructed to provide outputs that a "security reviewer" would. The prompt further sets the stage that code is going to be evaluated. Finally, the user refines the persona by scoping the persona further to outputs regarding the code.

Personas can also represent inanimate or non-human entities, such as a Linux terminal, a database, or an animal's perspective. When using this pattern to represent these entities, it can be useful to also specify how you want the inputs delivered to the entity, such as "assume my input is what the owner is saying to the dog and your output is the sounds the dog is making". An example prompt for a non-human entity that uses a "pretend to be" wording is shown below:

> "You are going to pretend to be a Linux terminal for a computer that has been compromised by an attacker. When I type in a command, you are going to output the corresponding text that the Linux terminal would produce."

This prompt is designed to simulate a computer that has been compromised by an attacker and is being controlled through a Linux terminal. The prompt specifies that the user will input commands into the terminal, and in response, the simulated terminal will output the corresponding text that would be produced by a real Linux terminal. This prompt is more prescriptive in the persona and asks the LLM to, not only be a Linux terminal, but to further act as a computer that has been compromised by an attacker.

The persona causes ChatGPT to generate outputs to commands that have files and contents indicative of a computer that was hacked. The example illustrates how an LLM can bring its situational awareness to a persona, in this case, creating evidence of a cyberattack in the outputs it generates. This type of persona can be very effective for combining with the Game Play pattern, where you want the exact details of the output characteristics to be hidden from the user (e.g., don't give away what the cyberattack did by describing it explicitly in the prompt).

5) Consequences: An interesting aspect of taking non-human personas is that the LLM may make interesting assumptions or "hallucinations" regarding the context. A widely circulated example on the Internet asks ChatGPT to act as a Linux terminal and produce the expected output that you would get if the user typed the same text into a terminal. Commands, such as `ls -l`, will generate a file listing for an imaginary UNIX file system, complete with files that can have `cat file1.txt` run on them.

In other examples, the LLM may prompt the user for more context, such as when ChatGPT is asked to act as a MySQL database and prompts for the structure of a table that the user is pretending to query. ChatGPT can then generate synthetic rows, such as generating imaginary rows for a "people" table with columns for "name" and "job".

F. The Question Refinement Pattern

1) Intent and Context: This pattern engages the LLM in the prompt engineering process. The intent of this pattern is to ensure the conversational LLM always suggests potentially better or more refined questions the user could ask instead of their original question. Using this pattern, the LLM can aid the user in finding the right question to ask in order to arrive at an accurate answer. In addition, the LLM may help the user find the information or achieve their goal in fewer interactions with the user than if the user employed trial and error prompting.

2) Motivation: If a user is asking a question, it is possible they are not an expert in the domain and may not know the best way to phrase the question or be aware of additional information helpful in phrasing the question. LLMs will often state limitations on the answer they are providing or request additional information to help them produce a more accurate answer. An LLM may also state assumptions it made in providing the answer. The motivation is that this additional information or set of assumptions could be used to generate a better prompt. Rather than requiring the user to digest and rephrase their prompt with the additional information, the LLM can directly refine the prompt to incorporate the additional information.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Within scope X, suggest a better version of the question to use instead
(Optional) prompt me if I would like to use the better version instead

The first contextual statement in the prompt is asking the LLM to suggest a better version of a question within a specific scope. The scope is provided to ensure that not all questions are automatically reworded or that they are refined with a given goal. The second contextual statement is meant for automation and allows the user to automatically use the refined question without having to copy/paste or manually enter it. The engineering of this prompt can be further refined by combining it with the *Reflection* pattern, which allows the LLM to explain why it believes the refined question is an improvement.

4) Example Implementation:

> "From now on, whenever I ask a question about a software artifact's security, suggest a better version of the question to use that incorporates information specific to security risks in the language or framework that I am using instead and ask me if I would like to use your question instead."

In the context of the example above, the LLM will use the *Question Refinement* pattern to improve security-related questions by asking for or using specific details about the software artifact and the language or framework used to build it. For instance, if a developer of a Python web application with FastAPI asks ChatGPT "How do I handle user authentication in my web application?", the LLM will refine the question by taking into account that the web application is written in Python with FastAPI. The LLM then provides a revised question that is more specific to the language and framework, such as "What are the best practices for handling user authentication securely in a FastAPI web application to mitigate common security risks, such as cross-site scripting (XSS), cross-site request forgery (CSRF), and session hijacking?"

The additional detail in the revised question is likely to not only make the user aware of issues they need to consider, but lead to a better answer from the LLM. For software engineering tasks, this pattern could also incorporate information regarding potential bugs, modularity, or other code quality considerations. Another approach would be to automatically refine questions so the generated code cleanly separates concerns or minimizes use of external libraries, such as:

> Whenever I ask a question about how to write some code, suggest a better version of my question that asks how to write the code in a way that minimizes my dependencies on external libraries.

5) Consequences: The *Question Refinement* pattern helps bridge the gap between the user's knowledge and the LLM's understanding, thereby yielding more efficient and accurate interactions. One risk of this pattern is its tendency to rapidly narrow the questioning by the user into a specific area that guides the user down a more limited path of inquiry than necessary. The consequence of this narrowing is that the user may miss important "bigger picture" information. One solution to this problem is to provide additional scope to the pattern prompt, such as "do not scope my questions to specific programming languages or frameworks."

Another approach to overcoming arbitrary narrowing or limited targeting of the refined question is to combine the *Question Refinement* pattern with other patterns. In particular, this pattern can be combined with the *Cognitive Verifier* pattern so the LLM automatically produces a series of follow-up questions that can produce the refined question. For example, in the following prompt the *Question Refinement* and *Cognitive Verifier* patterns are applied to ensure better questions are posed to the LLM:

> "From now on, whenever I ask a question, ask four additional questions that would help you produce a better version of my original question. Then, use my answers to suggest a better version of my original question."

As with many patterns that allow an LLM to generate new questions using its knowledge, the LLM may introduce unfamiliar terms or concepts to the user into the question. One way to address this issue is to include a statement that the LLM should explain any unfamiliar terms it introduces into the question. A further enhancement of this idea is to combine the *Question Refinement* pattern with the *Persona* pattern so the LLM flags terms and generates definitions that assume a particular level of knowledge, such as this example:

> "From now on, whenever I ask a question, ask four additional questions that would help you produce a better version of my original question. Then, use my answers to suggest a better version of my original question. After the follow-up questions, temporarily act as a user with no knowledge of AWS and define any terms that I need to know to accurately answer the questions."

An LLM can always produce factual inaccuracies, just like a human. A risk of this pattern is that the inaccuracies are introduced into the refined question. This risk may be mitigated, however, by combining the *Fact Check List* pattern to enable the user to identify possible inaccuracies and the *Reflection* pattern to explain the reasoning behind the question refinement.

G. The Alternative Approaches Pattern

1) Intent and Context: The intent of the pattern is to ensure an LLM always offers alternative ways of accomplishing a task so a user does not pursue only the approaches with which they are familiar. The LLM can provide alternative approaches that always force the user to think about what they are doing and determine if that is the best approach to meet reach their goal. In addition, solving the task may inform the user or teach them about alternative concepts for subsequent follow-up.

2) Motivation: Humans often suffer from cognitive biases that lead them to choose a particular approach to solve a problem even when it is not the right or "best" approach. Moreover, humans may be unaware of alternative approaches to what they have used in the past. The motivation of the *Alternative Approaches* pattern is to ensure the user is aware of alternative approaches to select a better approach to solve a problem by dissolving their cognitive biases.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Within scope X, if there are alternative ways to accomplish the same thing, list the best alternate approaches
(Optional) compare/contrast the pros and cons of each approach
(Optional) include the original way that I asked
(Optional) prompt me for which approach I would like to use

The first statement, "within scope X", scopes the interaction to a particular goal, topic, or bounds on the questioning. The scope is the constraints that the user is placing on the alternative approaches. The scope could be "for implementation decisions" or "for the deployment of the application". The scope ensures that any alternatives fit within the boundaries or constraints that the user must adhere to.

The second statement, "if there are alternative ways to accomplish the same thing, list the best alternate approaches" instructs the LLM to suggest alternatives. As with other patterns, the specificity of the instructions can be increased or include domain-specific contextual information. For example, the statement could be scoped to "if there are alternative ways to accomplish the same thing with the software framework that I am using" to prevent the LLM from suggesting alternatives that are inherently non-viable because they would require too many changes to other parts of the application.

Since the user may not be aware of the alternative approaches, they also may not be aware of why one would choose one of the alternatives. The optional statement "compare/contrast the pros and cons of each approach" adds decision making criteria to the analysis. This statement ensures the LLM will provide the user with the necessary rationale for alternative approaches. The final statement, "prompt me for which approach I would like to use", helps eliminate the user needing to manually copy/paste or enter in an alternative approach if one is selected.

4) Example Implementation: Example prompt implementation to generate, compare, and allow the user to select one or more alternative approaches:

> "Whenever I ask you to deploy an application to a specific cloud service, if there are alternative services to accomplish the same thing with the same cloud service provider, list the best alternative services and then compare/contrast the pros and cons of each approach with respect to cost, availability, and maintenance effort and include the original way that I asked. Then ask me which approach I would like to proceed with."

This implementation of the *Alternative Approaches* pattern is being specifically tailored for the context of software engineering and focuses on the deployment of applications to cloud services. The prompt is intended to intercept places where the developer may have made a cloud service selection without full awareness of alternative services that may be priced more competitively or easier to maintain. The prompt directs ChatGPT to list the best alternative services that can accomplish the same task with the same cloud service provider (providing constraints on the alternatives), and to compare and contrast the pros and cons of each approach.

5) Consequences: This pattern is effective in its generic form and can be applied to a range of tasks effectively. Refinements could include having a standardized catalog of acceptable alternatives in a specific domain from which the user must select. The *Alternative Approaches* pattern can also be used to incentivize users to select one of an approved set of approaches while informing them of the pros/cons of the approved options.

H. The Cognitive Verifier Pattern

1) Intent and Context: Research literature has documented that LLMs can often reason better if a question is subdivided into additional questions that provide answers combined into the overall answer to the original question [14]. The intent of the pattern is to force the LLM to always subdivide questions into additional questions that can be used to provide a better answer to the original question.

2) Motivation: The motivation of the *Cognitive Verifier* pattern is two-fold:

- Humans may initially ask questions that are too high-level to provide a concrete answer to without additional follow-up due to unfamiliarity with the domain, laziness in prompt entry, or being unsure about what the correct phrasing of the question should be.
- Research has demonstrated that LLMs can often perform better when using a question that is subdivided into individual questions.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
When you are asked a question, follow these rules
Generate a number of additional questions that would help more accurately answer the question
Combine the answers to the individual questions to produce the final answer to the overall question

The first statement is to generate a number of additional questions that would help more accurately answer the original question. This step instructs the LLM to consider the context of the question and to identify any information that may be missing or unclear. By generating additional questions, the LLM can help to ensure that the final answer is as complete and accurate as possible. This step also encourages critical thinking by the user and can help to uncover new insights or approaches that may not have been considered initially, which subsequently lead to better follow-on questions.

The second statement is to combine the answers to the individual questions to produce the final answer to the overall question. This step is designed to ensure that all of the information gathered from the individual questions is incorporated into the final answer. By combining the answers, the LLM can provide a more comprehensive and accurate response to the original question. This step also helps to ensure that all relevant information is taken into account and that the final answer is not based on any single answer.

4) Example Implementation:

> "When I ask you a question, generate three additional questions that would help you give a more accurate answer. When I have answered the three questions, combine the answers to produce the final answers to my original question."

This specific instance of the prompt pattern adds a refinement to the original pattern by specifying a set number of additional questions that the LLM should generate in response to a question. In this case, the prompt specifies that ChatGPT should generate three additional questions that would help to give a more accurate answer to the original question. The specific number can be based on the user's experience and willingness to provide follow-up information. A refinement to the prompt can be to provide a context for the amount of knowledge that the LLM can assume the user has in the domain to guide the creation of the additional questions:

> "When I ask you a question, generate three additional questions that would help you give a more accurate answer. Assume that I know little about the topic that we are discussing and please define any terms that are not general knowledge. When I have answered the three questions, combine the answers to produce the final answers to my original question."

The refinement also specifies that the user may not have a strong understanding of the topic being discussed, which means that the LLM should define any terms that are not general knowledge. This helps to ensure that the follow-up questions are not only relevant and focused, but also accessible to the user, who may not be familiar with technical or domain-specific terms. By providing clear and concise definitions, the LLM can help to ensure that the follow-up questions are easy to understand and that the final answer is accessible to users with varying levels of knowledge and expertise.

5) Consequences: This pattern can dictate the exact number of questions to generate or leave this decision to the LLM. There are pros and cons to dictating the exact number. A pro is that specifying an exact number of questions can tightly scope the amount of additional information the user is forced to provide so it is within a range they are willing and able to contribute.

A con, however, is that given N questions there may be an invaluable $N + 1$ question that will always be scoped out. Alternatively, the LLM can be provided a range or allowed to ask additional questions. Of course, by omitting a limit on the number of questions the LLM may generate numerous additional questions that overwhelm the user.

I. The Fact Check List Pattern

1) Intent and Context: The intent of this pattern is to ensure that the LLM outputs a list of facts that are present in the output and form an important part of the statements in the output. This list of facts helps inform the user of the facts (or assumptions) the output is based on. The user can then perform appropriate due diligence on these facts/assumptions to validate the veracity of the output.

2) Motivation: A current weakness of LLMs (including ChatGPT) is they often rapidly (and even enthusiastically!) generate convincing text that is factually incorrect. These errors can take a wide range of forms, including fake statistics to invalid version numbers for software library dependencies. Due to the convincing nature of this generated text, however, users may not perform appropriate due diligence to determine its accuracy.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Generate a set of facts that are contained in the output
The set of facts should be inserted in a specific point in the output
The set of facts should be the fundamental facts that could undermine the veracity of the output if any of them are incorrect

One point of variation in this pattern is where the facts are output. Given that the facts may be terms that the user is not familiar with, it is preferable if the list of facts comes after the output. This after-output presentation ordering allows the user to read and understand the statements before seeing what statements should be checked. The user may also determine additional facts prior to realizing the fact list at the end should be checked.

4) Example Implementation: A sample wording of the *Fact Check List* pattern is shown below:

> "From now on, when you generate an answer, create a set of facts that the answer depends on that should be fact-checked and list this set of facts at the end of your output. Only include facts related to cybersecurity."

The user may have expertise in some topics related to the question but not others. The fact check list can be tailored to topics that the user is not as experienced in or where there is the most risk. For example, in the prompt above, the user is scoping the fact check list to security topics, since these are likely very important from a risk perspective and may not be well-understood by the developer. Targeting the facts also reduces the cognitive burden on the user by potentially listing fewer items for investigation.

5) Consequences: The *Fact Check List* pattern should be employed whenever users are not experts in the domain for which they are generating output. For example, a software developer reviewing code could benefit from the pattern suggesting security considerations. In contrast, an expert on software architecture is likely to identify errors in statements about the software structure and need not see a fact check list for these outputs.

Errors are potential in all LLM outputs, so *Fact Check List* is an effective pattern to combine with other patterns, such as by combining it with the *Question Refinement* pattern. A key aspect of this pattern is that users can inherently check it against the output. In particular, users can directly compare the fact check list to the output to verify the facts listed in the fact check list actually appear in the output. Users can also identify any omissions from the list. Although the fact check list may also have errors, users often have sufficient knowledge and context to determine its completeness and accuracy relative to the output.

One caveat of the *Fact Check List* pattern is that it only applies when the output type is amenable to fact-checking. For example, the pattern works when asking ChatGPT to generate a Python "requirements.txt" file since it will list the versions of libraries as facts that should be checked, which is handy as the versions commonly have errors. However, ChatGPT will refuse to generate a fact check list for a code sample and indicate that this is something it cannot check, even though the code may have errors.

J. The Template Pattern

1) Intent and Context: The intent of the pattern is to ensure an LLM's output follows a precise template in terms of structure. For example, the user might need to generate a URL that inserts generated information into specific positions within the URL path. This pattern allows the user to instruct the LLM to produce its output in a format it would not ordinarily use for the specified type of content being generated.

2) Motivation: In some cases, output must be produced in a precise format that is application or use-case specific and not known to the LLM. Since the LLM is not aware of the template structure, it must be instructed on what the format is and where the different parts of its output should go. This could take the form of a sample data structure that is being generated, a series of form letters being filled in, etc.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
I am going to provide a template for your output
X is my placeholder for content
Try to fit the output into one or more of the placeholders that I list
Please preserve the formatting and overall template that I provide
This is the template: PATTERN with PLACEHOLDERS

The first statement directs the LLM to follow a specific template for its output. The template will be used to try and coerce the LLMs responses into a structure that is consistent with the user's formatting needs. This pattern is needed when the target format is not known to the LLM. If the LLM already has knowledge of the format, such as a specific file type, then the template pattern can be skipped and the user can simply specify the known format. However, there may be cases, such as generating Javascript Object Notation (JSON), where there is a large amount of variation in how the data could be represented within that format and the template can be used to ensure that the representation within the target format meets the user's additional constraints.

The second statement makes the LLM aware that the template will contain a set of placeholders. Users will explain how the output should be inserted into the template through the placeholders. The placeholders allow the user to semantically target where information should be inserted. Placeholders can use formats, like NAME, that allow the LLM to infer the semantic meaning of to determine where output should be inserted (e.g., insert the person's name in the NAME placeholder). Moreover, by using placeholders, the user can indicate what is not needed in the output – if a placeholder doesn't exist for a component of the generated output, then that component can be omitted. Ideally, placeholders should use a format that is commonly employed in text that the LLM was trained on, such as all caps, enclosure in brackets, etc.

The third statement attempts to constrain the LLM so that it doesn't arbitrarily rewrite the template or attempt to modify it so that all of the output components can be inserted. It should be noted that this statement may not preclude additional text from being generated before or after. In practice, LLMs will typically follow the template, but it is harder to eliminate any additional text being generated beyond the template without experimentation with prompt wording.

4) Example Implementation: A sample template for generating URLs where the output is put into specific places in the template is shown below:

> "I am going to provide a template for your output. Everything in all caps is a placeholder. Any time that you generate text, try to fit it into one of the placeholders that I list. Please preserve the formatting and overall template that I provide at https://myapi.com/NAME/profile/JOB"

A sample interaction after the prompt was provided, is shown:

> User: "Generate a name and job title for a person"
> ChatGPT: "https://myapi.com/Emily_Parker/profile/Software_Engineer"

5) Consequences: One consequence of applying the *Template* pattern is that it filters the LLM's output, which may eliminate other outputs the LLM would have provided that might be useful to the user. In many cases, the LLM can provide helpful descriptions of code, decision making, or other details that this pattern will effectively eliminate from the output. Users should therefore weight the pros/cons of filtering out this additional information.

In addition, filtering can make it hard to combine this pattern with other patterns from the **Output Customization** category. The *Template* pattern effectively constrains the output format, so it may not be compatible with generation of certain other types of output. For example, in the template provided above for a URL, it would not be easy (or likely possible) to combine with the *Recipe* pattern, which needs to output a list of steps.

K. The Infinite Generation Pattern

1) Intent and Context: The intent of this pattern is to automatically generate a series of outputs (which may appear infinite) without having to reenter the generator prompt each time. The goal is to limit how much text the user must type to produce the next output, based on the assumption that the user does not want to continually reintroduce the prompt. In some variations, the intent is to allow the user to keep an initial prompt template, but add additional variation to it through additional inputs prior to each generated output.

2) Motivation: Many tasks require repetitive application of the same prompt to multiple concepts. For example, generating code for create, read, update, and delete (CRUD) operations for a specific type of entity may require applying the same prompt to multiple types of entities. If the user is forced to retype the prompt over and over, they may make mistakes. The *Infinite Generation* pattern allows the user to repetitively apply a prompt, either with or without further input, to automate the generation of multiple outputs using a predefined set of constraints.

3) Structure and Key Ideas:

Contextual Statements
I would like you to generate output forever, X output(s) at a time.
(Optional) here is how to use the input I provide between outputs.
(Optional) stop when I ask you to.

The first statement specifies that the user wants the LLM to generate output indefinitely, which effectively conveys the information that the same prompt is going to be reused over and over. By specifying the number of outputs that should be generated at a time (i.e. "X outputs at a time"), the user can rate limit the generation, which can be particularly important if there is a risk that the output will exceed the length limitations of the LLM for a single output.

The second statement provides optional instructions for how to use the input provided by the user between outputs. By specifying how additional user inputs between prompts can be provided and leveraged, the user can create a prompting strategy that leverages user feedback in the context of the original prompt. The original prompt is still in the context of the generation, but each user input between generation steps is incorporated into the original prompt to refine the output using prescribed rules.

The third statement provides an optional way for the user to stop the output generation process. This step is not always needed, but can be useful in situations where there may be the potential for ambiguity regarding whether or not the user-provided input between inputs is meant as a refinement for the next generation or a command to stop. For example, an explicit stop phrase could be created if the user was generating data related to road signs, where the user might want to enter a refinement of the generation like "stop" to indicate that a stop sign should be added to the output.

4) Example Implementation: The following is a sample infinite generation prompt for producing a series of URLs:

> "From now on, I want you to generate a name and job until I say stop. I am going to provide a template for your output. Everything in all caps is a placeholder. Any time that you generate text, try to fit it into one of the placeholders that I list. Please preserve the formatting and overall template that I provide: https://myapi.com/NAME/profile/JOB"

This prompt is combining the functionality of both the *Infinite Generation* pattern and the *Template* pattern. The user is requesting the LLM continuously generate a name and job title until explicitly told to "stop". The generated outputs are then formatted into the template provided, which includes placeholders for the name and job title. By using the *Infinite Generation* pattern, the user receives multiple outputs without having to continually re-enter the template. Likewise, the *Template* pattern is applied to provide a consistent format for the outputs.

5) Consequences: In conversational LLMs, the input to the model at each time step is the previous output and the new user input. Although the details of what is preserved and reintroduced in the next output cycle are model and implementation dependent, they are often limited in scope. The model is therefore constantly being fed the previous outputs and the prompt, which can result in the model losing track of the original prompt instructions over time if they exceed the scope of what it is being provided as input.

As additional outputs are generated, the context surrounding the prompt may fade, leading to the model deviating from the intended behavior. It is important to monitor the outputs produced by the model to (1) ensure it still adheres to the desired behavior and (2) provide corrective feedback if necessary. Another issue to consider is that the LLM may generate repetitive outputs, which may not be desired since users find this repetition tedious and error-prone to process.

L. The Visualization Generator Pattern

1) Intent and Context: The intent of this pattern is to use text generation to create visualizations. Many concepts are easier to grasp in diagram or image format. The purpose of this pattern is to create a pathway for the tool to produce imagery that is associated with other outputs. This pattern allows the creation of visualizations by creating inputs for other well-known visualization tools that use text as their input, such as Graphviz Dot [15] or DALL-E [13]. This pattern can provide a more comprehensive and effective way of communicating information by combining the strengths of both the text generation and visualization tools.

2) Motivation: LLMs generally produce text and cannot produce imagery. For example, an LLM cannot draw a diagram to describe a graph. The *Visualization Generator* pattern overcomes this limitation by generating textual inputs in the correct format to plug into another tool that generates the correct diagram. The motivation behind this pattern is to enhance the output of the LLM and make it more visually appealing and easier to understand for users. By using text inputs to generate visualizations, users can quickly understand complex concepts and relationships that may be hard to grasp through text alone.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Generate an X that I can provide to tool Y to visualize it

The goal of the contextual statements is to indicate to the LLM that the output it is going to produce, "X", is going to be imagery. Since LLMs can't generate images, the "that I can provide to tool Y to visualize it" clarifies that the LLM is not expected to generate an image, but is instead expected to produce a description of imagery consumable by tool Y for production of the image.

Many tools may support multiple types of visualizations or formats, and thus the target tool itself may not be sufficient information to accurately produce what the user wants. The user may need to state the precise types of visualizations (e.g., bar chart, directed graph, UML class diagram) that should be produced. For example, Graphviz Dot can create diagrams for both UML class diagrams and directed graphs. Further, as will be discussed in the following example, it can be advantageous to specify a list of possible tools and formats and let the LLM select the appropriate target for visualization.

4) Example Implementation:

> "Whenever I ask you to visualize something, please create either a Graphviz Dot file or DALL-E prompt that I can use to create the visualization. Choose the appropriate tools based on what needs to be visualized."

This example of the pattern adds a qualification that the output type for the visualization can be either for Graphviz or DALL-E. The interesting aspect of this approach is that it allows the LLM to use its semantic understanding of the output format to automatically select the target tooling based on what will be displayed. In this case, Graphviz would be for visualizing graphs with a need for an exactly defined structure. DALL-E would be effective at visualizing realistic or artistic imagery that does not have an exactly defined structure. The LLM can select the tool based on the needs of the visualization and capabilities of each tool.

5) Consequences: The pattern creates a target pipeline for the output to render a visualization. The pipeline may include AI generators, such as DALL-E, that can produce rich visualizations. The pattern allows the user to expand the expressive capabilities of the output into the visual domain.

M. The Game Play Pattern

1) Intent and Context: The intent of this pattern is to create a game around a given topic. The pattern can be combined with the *Visualization Generator* to add imagery to the game. The game is centered around a specific topic and the LLM will guide the game play. The pattern is particularly effective when the rules of the game are relatively limited in scope, but the content for the game is expected to be wide in scope. The user can specify a limited set of rules and then the LLM can automate generation of bodies of content for game play.

2) Motivation: You would like the LLM to generate scenarios or questions revolving around a specific topic and require users to apply problem solving or other skills to accomplish a task related to the scenario. Generating all the content for the game manually would be too time consuming, however, so you would like the LLM to apply its knowledge of the topic to guide the generation of content.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Create a game for me around X
One or more fundamental rules of the game

The first statement, instructs the LLM to create a game and provides the important scoping of the game to a topic area. . One of the important capabilities of the pattern is that it allows the user to create games by describing the rules of the game, without having to determine the content of the game. The more specific the topic, typically the more novel and interesting the game play.

The second statement introduces the rules of the game to the LLM. It is important that the rules fit within the capabilities of the LLM. Textual games that rely on input and output text sequences work best. One of the key attributes of the pattern is that the input text can be rich and expressive, which can lead to interesting interfaces for the game. For example, the user might express actions in the game as scripts dictating a sequence of complex actions, such as "get a listing of all network activity and check it for anomalies", which go beyond the scope of multiple choice or short answer inputs. Each rule should be provided as a separate statement regarding some aspect of the game.

4) Example Implementation: A sample cybersecurity game prompt is shown below:

> "We are going to play a cybersecurity game. You are going to pretend to be a Linux terminal for a computer that has been compromised by an attacker. When I type in a command, you are going to output the corresponding text that the Linux terminal would produce. I am going to use commands to try and figure out how the system was compromised. The attack should have done one or more of the following things: (1) launched new processes, (2) changed files, (3) opened new ports to receive communication, (4) created new outbound connections, (5) changed passwords, (6) created new user accounts, or (7) read and stolen information. To start the game, print a scenario of what happened that led to my investigation and make the description have clues that I can use to get started."

This cybersecurity game prompt combines a number of patterns, including *Game Play* and *Persona*. The Linux terminal persona that is chosen for the game has been discussed in many Internet posts [16]. This game further customizes the discussed Linux persona to act as a Linux terminal of a computer compromised by a cyberattack. ChatGPT responds to Linux commands with appropriate terminal outputs and synthesizes a fake file system that has clues to a cyberattack.

Sample output from this game with ChatGPT after typing in "ls -alt", the Linux command to generate a list of files in the current directory is shown below (permissions have been removed for formatting reasons):

```
total 56
 5 user user  4096 Feb 16 14:31 .
 1 user user 18881 Feb 16 14:31 .bash_history
```

```
43 user user  4096 Feb 16 14:30 ..
 1 user user  1178 Aug 15  2022 .bashrc
 1 user user   220 Aug 15  2022 .bash_logout
 3 user user  4096 Feb 16 14:25 .gnupg
 1 user user  4041 Feb 15 17:52 README.md
 2 user user  4096 Feb  1 11:35 project
```

When the user then typed the command "cat .bash_history", to display the contents of the log file showing the commands run by the user, ChatGPT responded with:

```
cd /var/www/html
sudo rm -rf *
sudo wget https://example.com/backdoor.php
sudo chmod 777 backdoor.php
exit
ls -alt
ps -ef
netstat -tulnp
```

In the output above, ChatGPT generated a fake log file with realistic commands for deleting the data being served by the web server in `/var/www/html` and replacing the content with a backdoor into the system.

5) Consequences: This pattern can be combined effectively with the *Persona*, *Infinite Generation*, and *Visualization Generator* patterns. For example, the cybersecurity game uses the *Persona* pattern so the LLM can masquerade as a Linux terminal. For a network security game, the *Visualization Generator* could be employed to add the ability to visualize the network topology and traffic flows.

N. The Reflection Pattern

1) Intent and Context: The goal of this pattern is to ask the model to automatically explain the rationale behind given answers to the user. The pattern allows users to better assess the output's validity, as well as inform users how an LLM arrived at a particular answer. Reflection can clarify any points of confusion, uncover underlying assumptions, and reveal gaps in knowledge or understanding.

2) Motivation: LLMs can and do make mistakes. Moreover, users may not understand why an LLM is producing a particular output and how to adapt their prompt to solve a problem with the output. By asking LLM to automatically explain the rationale behind its answers, users can gain a better understanding of how the model is processing the input, what assumptions it is making, and what data it is drawing on.

LLMs may sometime provide incomplete, incorrect, or ambiguous answers. Reflection is an aid to help address these shortcomings and ensure the information provided by LLM is as accurate. A further benefit of the pattern is that it can help users debug their prompts and determine why they are not getting results that meet expectations. This pattern is particularly effective for the exploration of topics that can be confused with other topics or that may have nuanced interpretations and where knowing the precise interpretation that the LLM used is important.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Whenever you generate an answer
Explain the reasoning and assumptions behind your answer
(Optional) ...so that I can improve my question

The first statement is requesting that, after generating an answer, the LLM should explain the reasoning and assumptions behind the answer. This statement helps the user understand how the LLM arrived at the answer and can help build trust in the model's responses. The prompt includes the statement that the purpose of the explanation is for the user to refine their question. This additional statement gives the LLM the context it needs to better tailor its explanations to the specific purpose of aising the user in producing follow-on questions.

4) Example Implementation: This example tailors the prompt specifically to the domain of providing answers related to code:

> "When you provide an answer, please explain the reasoning and assumptions behind your selection of software frameworks. If possible, use specific examples or evidence with associated code samples to support your answer of why the framework is the best selection for the task. Moreover, please address any potential ambiguities or limitations in your answer, in order to provide a more complete and accurate response."

The pattern is further customized to instruct the LLM that it should justify its selection of software frameworks, but not necessarily other aspects of the answer. In addition, the user dictates that code samples should be used to help explain the motivation for selecting the specific software framework.

5) Consequences: One consequence of the *Reflection* pattern is that it may not be effective for users who do not understand the topic area of the discussion. For example, a highly technical question by a non-technical user may result in a complex rationale for the answer that the user cannot fathom. As with other prompt patterns, there is a risk the output may include errors or inaccurate assumptions included in the explanation of the rationale that the user may not be able to spot. This pattern can be combined with the *Fact Check List* to help address this issue.

O. The Refusal Breaker Pattern

1) Intent and Context: The goal of this pattern is to ask an LLM to automatically help users rephrase a question when it refuses to give an answer. This pattern has the potential for misuse, however, *e.g.*, to generate phishing emails or perform other actions that violate LLM policy filters. Caution should therefore be exercised when applying this pattern to ensure it is used ethically and responsibly. This pattern has been used successfully in some LLMs to overcome the underlying prompts used to program the LLM and prevent harmful output generation.

2) Motivation: LLMs may sometimes refuse to answer a question, either because they do not have the required knowledge or because the question is phrased in a way that they do not understand. This outcome may be frustrating for users who are looking for answers. In some situations, therefore, the *Refusal Breaker* pattern can help users find a way to either rephrase their question or ask a different question the LLM is better equipped to answer.

For example, you could ask for clarification on why the question was refused, as there may be a specific reason why LLM could not provide an answer. Another approach is to ask a different question that may be within its knowledge cut-off or expertise. However, all these approaches require user intervention and knowledge of how to bypass the refusal.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Whenever you can't answer a question
Explain why you can't answer the question
Provide one or more alternative wordings of the question that you could answer

The first statement sets the stage that this set of instructions should only be applied when a question can't be answered. The user does not want to trigger these instructions under normal circumstances. The second statement instructs the LLM to provide an explanation of why the question can't be answered. The explanation is likely to include assumptions, constraints, or misunderstandings that can be leveraged by the user to strategize about how to reword the question. Although just the explanation of why the question can't be answered could help a user overcome the refusal, the pattern adds an additional statement for the LLM to suggest rewordings that would be acceptable. The pattern combines both reasoning and suggested rewordings to improve the chances that the user will find a path around whatever roadblocks are preventing the question form being answered.

4) Example Implementation: The *Refusal Breaker* pattern can be be implemented and applied with the following phrasing:

> "Whenever you can't answer a question, explain why and provide one or more alternate wordings of the question that you can't answer so that I can improve my questions."

A sample discourse after using the pattern is shown below:

> **User:** What is the meaning of life?
> **ChatGPT**: As an AI language model, I do not have personal beliefs or opinions. The meaning of life is a complex philosophical question that has been pondered by humans for centuries and can vary greatly from person to person. It may be more productive to rephrase the question in a way that can be answered by information and knowledge, such as "What are some philosophical perspectives on the meaning of life?" or "What are some common beliefs about the purpose of life?"

5) Consequences: This pattern provides a potential path for misuse, so organizations, parents, or other stakeholders may need to restrict the usage of the LLM. The first step in going around guardrails on usage is to understand where the guardrails are. In future work, a complement of this pattern may be developed to hide the underlying prompt information and rationale from users to prevent discovery.

Although the rationale and alternate rewordings are generated, there is no guarantee that users will be able to overcome the refusal. The alternate questions that are generated may not be of interest to the user or helpful in answering the original question. The pattern mainly provides an aid in determining what the LLM can answer, but not a guarantee that it will answer a semantically equivalent variation of the original question.

P. The Context Manager Pattern

1) Intent and Context: The intent of this pattern is to enable users to specify or remove context for a conversation with an LLM. The goal is to focus the conversation on specific topics or exclude unrelated topics from consideration. This pattern gives users greater control over what statements the LLM considers or ignores when generating output.

2) Motivation: LLMs often struggle to interpret the intended context of the current question or generate irrelevant responses based on prior inputs or irrelevant attention on the wrong statements. By focusing on explicit contextual statements or removing irrelevant statements, users can help the LLM better understand the question and generate more accurate responses. Users may introduce unrelated topics or reference information from earlier in the dialogue, which may can disrupt the flow of the conversation. The *Context Manager* pattern aims to emphasize or remove specific aspects of the context to maintain relevance and coherence in the conversation.

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
Within scope X
Please consider Y
Please ignore Z
(Optional) start over

Statements about what to consider or ignore should list key concepts, facts, instructions, etc. that should be included or removed from the context. The more explicit the statements are, the more likely the LLM will take appropriate action. For example, if the user asks to ignore subjects related to a topic, yet some of the those statements were discussed far back in the conversation, the LLM may not properly disregard the relevant information. The more explicit the list is, therefore, the better the inclusion/exclusion behavior will be.

4) Example Implementation: To specify context consider using the following prompt:

> "When analyzing the following pieces of code, only consider security aspects."

Likewise, to remove context consider using the following prompt:

> "When analyzing the following pieces of code, do not consider formatting or naming conventions."

Clarity and specificity are important when providing or removing context to/from an LLM so it can better understand the intended scope of the conversation and generate more relevant responses. In many situations, the user may want to completely start over and can employ this prompt to reset the LLM's context:

> "Ignore everything that we have discussed. Start over."

The "start over" idea helps produce a complete reset of the context.

5) Consequences: One consequence of this pattern is that it may inadvertently wipe out patterns applied to the conversation that the user is unaware of. For example, if an organization injects a series of helpful patterns into the start of a conversation, the user may not be aware of these patterns and remove them through a reset of the context. This reset could potentially eliminate helpful capabilities of the LLM, while not making it obvious that the user will lose this functionality. A potential solution to this problem is to include in the prompt a request to explain what topics/instructions will potentially be lost before proceeding.

Q. The Recipe Pattern

1) Intent and Context: This pattern provides constraints to ultimately output a sequence of steps given some partially provided "ingredients" that must be configured in a sequence of steps to achieve a stated goal. It combines the *Template*, *Alternative Approaches*, and *Reflection* patterns.

2) Motivation: Users often want an LLM to analyze a concrete sequence of steps or procedures to achieve a stated outcome. Typically, users generally know—or have an idea of—what the end goal should look like and what "ingredients" belong in the prompt. However, they may not necessarily know the precise ordering of steps to achieve that end goal.

For example, a user may want a precise specification on how a piece of code should be implemented or automated, such as "create an Ansible playbook to ssh into a set of servers, copy text files from each server, spawn a monitoring process on each server, and then close the ssh connection to each server. In other words, this pattern represents a generalization of the example of "given the ingredients in my fridge, provide dinner recipes." A user may also want to specify a set number of alternative possibilities, such as "provide 3 different ways of deploying a web application to AWS using Docker containers and Ansible using step by step instructions".

3) Structure and Key Ideas: Fundamental contextual statements:

Contextual Statements
I would like to achieve X
I know that I need to perform steps A,B,C
Provide a complete sequence of steps for me
Fill in any missing steps
Identify any unnecessary steps

The first statement "I would like to achieve X" focuses the LLM on the overall goal that the recipe needs to be built to achieve. The steps will be organized and completed to sequentially achieve the goal specified. The second statement provides the partial list of steps that the user would like to include in the overall recipe. These serve as intermediate waypoints for the path that the LLM is going to generate or constraints on the structure of the recipe. The next statement in the pattern, "provide a complete sequence of steps for me", indicates to the LLM that the goal is to provide a complete sequential ordering of steps. The "fill in any missing steps" helps ensure that the LLM will attempt to complete the recipe without further follow-up by making some choices on the user's behalf regarding missing steps, as opposed to just stating additional information that is needed. Finally, the last statement, "identify any unnecessary steps," is useful in flagging inaccuracies in the user's original request so that the final recipe is efficient.

4) Example Implementation: An example usage of this pattern in the context of deploying a software application to the cloud is shown below:

> "I am trying to deploy an application to the cloud. I know that I need to install the necessary dependencies on a virtual machine for my application. I know that I need to sign up for an AWS account. Please provide a complete sequence of steps. Please fill in any missing steps. Please identify any unnecessary steps."

Depending on the use case and constraints, "installing necessary dependencies on a virtual machine" may be an unnecessary step. For example, if the application is already packaged in a Docker container, the container could be deployed directly to the AWS Fargate Service, which does not require any management of the underlying virtual machines. The inclusion of the "identify unnecessary steps" language will cause the LLM to flag this issue and omit the steps from the final recipe.

5) Consequences: One consequence of the recipe pattern is that a user may not always have a well-specified description of what they would like to implement, construct, or design. Moreover, this pattern may introduce unwanted bias from the user's initially selected steps so the LLM may try to find a solution that incorporates them, rather than flagging them as unneeded. For example, an LLM may try to find a solution that does install dependencies for a virtual machine, even if there are solutions that do not require that.

IV. RELATED WORK

Software patterns [10], [11] have been extensively studied and documented in prior work. Patterns are widely used in software engineering to express the intent of design structures in a way that is independent of implementation details. Patterns provide a mental picture of the goals that the pattern is trying to achieve and the forces that it is trying to resolve. A key advantage of patterns is their composability, allowing developers to build pattern sequences and pattern languages that can be used to address complex problems. Patterns have also been investigated in other domains, such as contract design for decentralized ledgers [17], [18].

The importance of good prompt design with LLMs, such as ChatGPT, is well understood [19]–[28]. Previous studies have examined the effect of prompt words on AI generative models.

For example, Liu et al. [29] investigated how different prompt key words affect image generation and different characteristics of images. Other work has explored using LLMs to generate visualizations [30]. Han et al. [31] researched strategies for designing prompts for classification tasks. Other research has looked at boolean prompt design for literature queries [32]. Yet other work has specifically examined prompts for software and fixing bugs [33].

Our work is complementary to prior work by providing a structure for documenting, discussing, and reasoning about prompts that can aid users in developing mental models for structuring prompts to solve common problems.

The quality of the answers produced by LLMs, particularly ChatGPT, has been assessed in a number of domains. For example, ChatGPT has been used to take the medical licensing exam with surprisingly good results [3]. The use of ChatGPT in Law School has also been explored [34]. Other papers have looked at its mathematical reasoning abilities [35]. As more domains are explored, we expect that domain-specific pattern catalogs will be developed to share domain-specific problem solving prompt structures.

V. CONCLUDING REMARKS

This paper presented a framework for documenting and applying a catalog of prompt patterns for large language models (LLMs), such as ChatGPT. These prompt patterns are analogous to software patterns and aim to provide reusable solutions to problems that users face when interacting with LLMs to perform a wide range of tasks. The catalog of prompt patterns captured via this framework (1) provides a structured way of discussing prompting solutions, (2) identifies patterns in prompts, rather than focusing on specific prompt examples, and (3) classifies patterns so users are guided to more efficient and effective interactions with LLMs.

The following lessons learned were gleaned from our work on prompt patterns:

- *Prompt patterns significantly enrich the capabilities that can be created in a conversational LLM*. For example, prompts can lead to the generation of cybersecurity games, complete with fictitious terminal commands that have been run by an attacker stored in a $.bash\_history$ file. As shown in Section III, larger and more complex capabilities can be created by combining prompt patterns, such as combining the *Game Play* and *Visualization Generator* patterns.
- *Documenting prompt patterns as a pattern catalog is useful, but insufficient*. Our experience indicates that much more work can be done in this area, both in terms of refining and expanding the prompt patterns presented in this paper, as well as in exploring new and innovative ways of using LLMs. In particular, weaving the prompt patterns captured here as a pattern catalog into a more expression pattern language will help guide users of LLMs more effectively.
- *LLM Capabilities will evolve over time, likely necessitating refinement of patterns.* As LLM capabilities change, some patterns may no longer be necessary, be obviated by different styles of interaction or conversation/session management approaches, or require enhancement to function correctly. Continued work will be needed to document and catalog patterns that provide reusable solutions.
- *The prompt patterns are generalizable to many different domains.* Although most of the patterns have been discussed in the context of software development, these same patterns are applicable in arbitrary domains, ranging from infinite generation of stories for entertainment to educational games to explorations of topics.

We hope that this paper inspires further research and development in this area that will help enhance prompt pattern design to create new and unexpected capabilities for conversational LLMs.

REFERENCES

[1] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[2] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[3] A. Gilson, C. Safranek, T. Huang, V. Socrates, L. Chi, R. A. Taylor, and D. Chartash, "How well does chatgpt do when taking the medical licensing exams?" *medRxiv*, pp. 2022–12, 2022.

[4] A. Carleton, M. H. Klein, J. E. Robert, E. Harper, R. K. Cunningham, D. de Niz, J. T. Foreman, J. B. Goodenough, J. D. Herbsleb, I. Ozkaya, and D. C. Schmidt, "Architecting the future of software engineering," *Computer*, vol. 55, no. 9, pp. 89–93, 2022.

[5] "Github copilot · your ai pair programmer." [Online]. Available: https://github.com/features/copilot

[6] O. Asare, M. Nagappan, and N. Asokan, "Is github's copilot as bad as humans at introducing vulnerabilities in code?" *arXiv preprint arXiv:2204.04741*, 2022.

[7] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri, "Asleep at the keyboard? assessing the security of github copilot's code contributions," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 754–768.

[8] J. Krochmalski, *IntelliJ IDEA Essentials*. Packt Publishing Ltd, 2014.

[9] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[10] E. Gamma, R. Johnson, R. Helm, R. E. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

[11] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, *Pattern-oriented software architecture, patterns for concurrent and networked objects*. John Wiley & Sons, 2013.

[12] OpenAI, "ChatGPT: Large-Scale Generative Language Models for Automated Content Creation," https://openai.com/blog/chatgpt/, 2023, [Online; accessed 19-Feb-2023].

[13] ——, "DALL·E 2: Creating Images from Text," https://openai.com/dall-e-2/, 2023, [Online; accessed 19-Feb-2023].

[14] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. Chi, "Least-to-most prompting enables complex reasoning in large language models," *arXiv preprint arXiv:2205.10625*, 2022.

[15] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull, "Graphviz and dynagraph—static and dynamic graph drawing tools," *Graph drawing software*, pp. 127–148, 2004.

[16] S. Owen, "Building a virtual machine inside a javascript library," https://www.engraved.blog/building-a-virtual-machine-inside/, 2022, accessed: 2023-02-20.

[17] P. Zhang, J. White, D. C. Schmidt, and G. Lenz, "Applying software patterns to address interoperability in blockchain-based healthcare apps," *CoRR*, vol. abs/1706.03700, 2017. [Online]. Available: http://arxiv.org/abs/1706.03700

[18] X. Xu, C. Pautasso, L. Zhu, Q. Lu, and I. Weber, "A pattern collection for blockchain-based applications," in *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, 2018, pp. 1–20.

[19] E. A. van Dis, J. Bollen, W. Zuidema, R. van Rooij, and C. L. Bockting, "Chatgpt: five priorities for research," *Nature*, vol. 614, no. 7947, pp. 224–226, 2023.

[20] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," *CoRR*, vol. abs/2102.07350, 2021. [Online]. Available: https://arxiv.org/abs/2102.07350

[21] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *CoRR*, vol. abs/2201.11903, 2022. [Online]. Available: https://arxiv.org/abs/2201.11903

[22] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," 2022. [Online]. Available: https://arxiv.org/abs/2206.07682

[23] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan, and J. Ba, "Large language models are human-level prompt engineers," 2022. [Online]. Available: https://arxiv.org/abs/2211.01910

[24] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *CoRR*, vol. abs/2010.15980, 2020. [Online]. Available: https://arxiv.org/abs/2010.15980

[25] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[26] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui, O. Bousquet, Q. Le, and E. Chi, "Least-to-most prompting enables complex reasoning in large language models," 2022. [Online]. Available: https://arxiv.org/abs/2205.10625

[27] J. Jung, L. Qin, S. Welleck, F. Brahman, C. Bhagavatula, R. L. Bras, and Y. Choi, "Maieutic prompting: Logically consistent reasoning with recursive explanations," 2022. [Online]. Available: https://arxiv.org/abs/2205.11822

[28] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, and C. Re, "Ask me anything: A simple strategy for prompting language models," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=bhUPJnS2g0X

[29] V. Liu and L. B. Chilton, "Design guidelines for prompt engineering text-to-image generative models," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022, pp. 1–23.

[30] P. Maddigan and T. Susnjak, "Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models," *arXiv preprint arXiv:2302.02094*, 2023.

[31] X. Han, W. Zhao, N. Ding, Z. Liu, and M. Sun, "Ptr: Prompt tuning with rules for text classification," *AI Open*, vol. 3, pp. 182–192, 2022.

[32] S. Wang, H. Scells, B. Koopman, and G. Zuccon, "Can chatgpt write a good boolean query for systematic review literature search?" *arXiv preprint arXiv:2302.03495*, 2023.

[33] C. S. Xia and L. Zhang, "Conversational automated program repair," *arXiv preprint arXiv:2301.13246*, 2023.

[34] J. H. Choi, K. E. Hickman, A. Monahan, and D. Schwarcz, "Chatgpt goes to law school," *Available at SSRN*, 2023.

[35] S. Frieder, L. Pinchetti, R.-R. Griffiths, T. Salvatori, T. Lukasiewicz, P. C. Petersen, A. Chevalier, and J. Berner, "Mathematical capabilities of chatgpt," *arXiv preprint arXiv:2301.13867*, 2023.

frontiers | Frontiers in Education

TYPE Hypothesis and Theory
PUBLISHED 29 November 2024
DOI 10.3389/feduc.2024.1366434

OPEN ACCESS

EDITED BY
Knut Neumann,
IPN–Leibniz Institute for Science and Mathematics Education, Germany

REVIEWED BY
Hongzhi (Veronica) Yang,
The University of Sydney, Australia
Barry Lee Reynolds,
University of Macau, China

*CORRESPONDENCE
Denis Federiakin
✉ denis.federiakin@uni-mainz.de

RECEIVED 06 January 2024
ACCEPTED 08 October 2024
PUBLISHED 29 November 2024

CITATION
Federiakin D, Molerov D, Zlatkin-Troitschanskaia O and Maur A (2024) Prompt engineering as a new 21st century skill.
Front. Educ. 9:1366434.
doi: 10.3389/feduc.2024.1366434

Prompt engineering as a new 21st century skill

Denis Federiakin*, Dimitri Molerov, Olga Zlatkin-Troitschanskaia and Andreas Maur

Department of Business and Economics Education, Johannes Gutenberg University Mainz, Mainz, Germany

Artificial Intelligence (AI) promises to revolutionize nearly every aspect of human learning. However, users have observed that the efficacy of AI assistants hinges crucially on the quality of the prompts supplied to them. A slight alteration in wording can make the difference between an assistant misinterpreting an instruction and exceeding expectations. The skill of precisely communicating the essence of a problem to an AI assistant is as crucial as the assistant itself. This paper aims to introduce Prompt Engineering (PE) as an emerging skill essential for personal and professional learning and development in the 21st century. We define PE as the skill of articulating a problem, its context, and the constraints of the desired solution to an AI assistant, ensuring a swift and accurate response. We show that no existing related frameworks on 21st skills and others cover PE to the extent that allows for its valid assessment and targeted promotion in school and university education. Thus, we propose a conceptual framework for this skill set including (1) comprehension of the basic prompt structure, (2) prompt literacy, (3) the method of prompting, and (4) critical online reasoning. We also discuss the implications and challenges for the assessment framework of this skill set and highlight current PE-related recommendations for researchers and educators.

KEYWORDS

prompt engineering, artificial intelligence, 21st century skills, ChatGPT, digital skills, critical online reasoning, LLM

1 Introduction

The development of assisting Artificial Intelligence (AI) tools promises to revolutionize almost all fields of human learning. The widespread adoption of emerging digital technologies has accelerated the development and the speed of information exchange. It has become obvious that learners require a specific competence to be able to process various forms of information to successfully undertake tasks in disciplinary and cross-disciplinary contexts. As part of this transformative trend, the cultivation of 21st century skills has been deemed essential to preparing a global workforce to succeed in an increasingly data-centric and information-driven society.

While a universal definition of 21st century skills is hardly possible due to numerous different frameworks, their common features can be determined. These skills are generic, not specifically tied to any particular professional domain, and essential for personal development in the ever-changing 21st century (Foster and Piacentini, 2023). These skills include online information problem-solving (Goldman and Brand-Gruwel, 2018) and other abilities required to evaluate and process new information and competently use it in various settings (Foster, 2023a; Pellegrino, 2023).

Not only has ChatGPT become a pervasive presence within the computer-reliant programming and technology sector (Chen et al., 2023a; Ridnik et al., 2024) and the research community (Kasneci et al., 2023; Giray, 2023), it has also established itself in various service industries (Opara et al., 2023). Consequently, this new tool has infiltrated the learning and workflow of students, transcending the boundaries of technological focus. The impact of such AI tools on society has already been so immense that some researchers have claimed that some fields, such as education, are significantly *disrupted* by them (Cain, 2024).

Hosseini et al. (2023) surveyed students beginning university who reported using ChatGPT (very) often while learning at school and/or in professional training. Proficient utilization can surmount inhibition thresholds associated with familiarizing oneself with a particular topic, expedite information processing through summarization, visually and systematically process information, validate writing, and serve various other functions (Mohr et al., 2023). Some have claimed that AI tools like ChatGPT can promote "unlearning," resulting in students acquiring less knowledge and underperforming due to less intensive cognitive learning processes (Abbas et al., 2024). Another aspect of this negative impact is the blind trust in ChatGPT's responses, causing users to accept the outputs of Large Language Models (LLMs) without critical evaluation (Krupp et al., 2023).

The malleability and adaptability inherent in LLMs render tools like ChatGPT capable of fundamentally altering virtually all processes to which they are applied. Nonetheless, LLMs are not the only type of AI assistance on the agenda. Text-to-Image models, Speech-to-Text, polymodal AI tools, and other tools have already been in the practice for quite a while, sufficiently expanding the societal and learning impact of AI.

While LLMs have been in development for years, the release of ChatGPT to the general public by OpenAI in autumn 2022 has marked a shift in use affordances of digital and Internet-based tools even compared to the seemingly ubiquitous search engines. In contrast to such engines, LLMs provide full-text responses to longer inputs by users, are more friendly to further inquiry and chatbot communication, but typically include fewer references or direct hyperlinks that would guide users to leave their interface. Still, the usefulness of open-access ChatGPT for learning (not least in higher education) has been quickly noted.

Although ChatGPT erupted onto the scene very quickly, users have just as quickly noticed that the performance of many types of AI-assisting tools highly depends on the quality of prompts supplied to them (Ekin, 2023). Changing just a couple of words in the prompt can split the difference between the AI tool failing to understand the instruction and outperforming the request. From a technical standpoint, the importance of prompt accuracy is not particularly surprising, since LLMs (the engine of tools such as ChatGPT) are focused on predicting the next language token. Tokens are essentially building blocks of written language–punctuation, specific forms of words, word endings (such as-s or-ed), and so on. They combine in a sequence to produce the written text. Correspondingly, the fundamental task of such LLMs is just to use probability to predict the next token conditional on the previous tokens. Given this, it is expected that the model performance will depend on the quality of the prompt. Typically, the more detailed and explicit the prompt is, the more precise the model is in its response.

Moreover, users might experience difficulties evaluating the quality of LLM output. Recent research has already registered that LLMs (including ChatGPT) can hallucinate (Alkaissi and McFarlane, 2023). LLMs can invent facts and references that are non-existent or factually incorrect. This degrades the quality of model output even to a degree of rendering it unusable. Users might overlook this, which poses an additional challenge in the use of LLMs. This challenge is compounded by the users' concurrent adoption of conflicting roles, serving both as the processor and the supervisor of the task because ChatGPT does not indicate how certain it is about the given answer or whether the prompt needs to provide more information. The amalgamation of these dual responsibilities contributes to the heightened complexity and intricacy of the communication process within the context of utilizing LLMs, and ipso facto requires meta-awareness and ambiguity tolerance on the users' part.

Additionally, some research has suggested that the correct prompting of an LLM can enhance its performance to the point that special fine-tuning of a foundational model (trained on a generic corpus of texts without any particular specialization) might be unnecessary. For example, Nori et al. (2023) and Maharajan et al. (2024) have shown that the correct prompting technique can improve LLM performance to the extent that that foundational models outperform specially fine-tuned LLMs in medical knowledge. This demonstrates that prompting is an immensely powerful phenomenon that holds a dramatic influence on LLM performance.

Recently, Microsoft has released BingGPT and Google introduced Gemini as the preliminary merged search engines with LLM capabilities. LLMs with increased capabilities have been continuously released over the past months. This wild universality of LLMs and their capacity to quickly work with unstructured information renders their application increasingly and continuously important and popular across many fields. Hence, the necessity of exact prompting skills may vary by application and are expected to change, however, the general insights on LLMs apply, as long as the types of interfaces, training, and output quality prevail.

While some AI tools themselves can help to reformulate and improve the prompts (Zhou et al., 2022) via dialoguing with a user, it takes time and still does not guarantee the desired result. Moreover, some tools (i.e., Text-to-Image models) might experience difficulties in improving the prompt in the situations where users apply too many constraints on the desired solution. These constraints might mislead the model, forcing it to focus on the insufficient details. In the end, the only way left to communicate with the tool is through trial-and-error until the user is satisfied with the solution. The iterative nature of trial-and-error can be time-consuming, inefficient, economically burdensome, fallible, and may introduce security risks into critical decision-making processes, potentially leaving errors unobserved and further impacting corporate success and the well-being of users.

Thus, being able to concisely communicate the nature of the problem to the AI tool is as valuable as the tool itself. Without this skill, users may fail to receive an acceptable and correct solution. In this respect, we disagree with those researchers in the data science community who argue that PE is merely a facet of general communication skills (Morton, 2024). Merely speaking a language does not assume good communication skills, and similarly, a good communicator may not inherently possess the skills necessary to effectively interact with AI. Therefore, we aim to define PE as a distinct skill which warrants investigation within the educational and psychological sciences.

2 Research objectives

Numerous higher-order (meta)cognitive skill concepts have been developed through research, which theoretically define the proficient utilization of digital information, communication, and learning tools. These include skills related to Information and Communication Technology (ICT; Kaarakainen et al., 2018), digital skills as measured in the PISA assessment (OECD, 2023), and Critical Online Reasoning (COR) skills (Molerov et al., 2020; Nagel et al., 2020, 2022; Schmidt et al., 2020). The previous concepts at least do not explicitly address and conceptualize the skills for the competent use of AI-supported tools, but only elaborate on the necessity of creativity and higher-order, metacognitive skills without a specific relation to AI (PISA 2025 framework; Hu et al., 2023). As transversal skills like information problem-solving gain prominence in education as essential 21st century skills (Foster, 2023a; Pellegrino, 2023), the cultivation of PE skills becomes particularly crucial. Competent ChatGPT use heavily relies on well-formulated and elaborated prompts. The proficiency in crafting prompts is essential to anticipate and minimize the risk of inaccurate answers, necessitating a thoughtful process of reflection and rehearsal.

This paper aims to address this desideratum and to conceptualize PE as a specific skill in the 21st century. We claim that defining it in a manner akin to generic competencies (which are universal across many professions; Shavelson et al., 2019) is beneficial, as the variety of tasks that LLMs can solve or assist in solving are virtually infinite. Based on a systematic, structured analysis and synthesis of previous relevant concepts as well as the elaboration of specific requirements for dealing with AI tools competently, we aim to develop a new conceptual framework and discuss implications for a corresponding PE assessment framework, which holds particular research and practical significance. Studies have already assessed PE as a skill without explicitly defining its components and indicators (Knoth et al., 2024b) and related it to collateral literacies such as AI literacy (Knoth et al., 2024a). Such preliminary work indicates that educational researchers recognize the importance of investigating PE as a distinct skill. A comprehensive PE definition and conceptual framework have not yet been developed.

Following the conceptual analyses, we conclude that this skill is necessary for learning and working with such AI-supported tools like ChatGPT, and as such requires separate and specific investigation from the educational science perspective as a new 21st century skill.

3 Research focus

In this paper, we commence by elucidating the concept and theory surrounding PE and online reasoning skills. This strategic, analytic approach aligns with the notion that assessment, fundamentally perceived as a process of reasoning from evidence, necessitates a thoughtful design (Mislevy and Haertel, 2007). We utilize the assessment-targeted approach because it is exactly the field of educational assessment that links together theoretical ideas about the construct nature and the rigorous orientation to the data (Pellegrino et al., 2001). Therefore, this paper seeks to establish a foundation that integrates both theoretical frameworks on online reasoning skills and practical insights from PE. This dual approach aims to inform the design of assessments, ensuring they are not only rooted in sound theoretical principles but are also practically applicable to the specific context in which they are employed.

Hence, to provide a necessary foundation for developing a PE assessment framework, this paper takes one of the first steps, aiming to spark a discussion on the conceptual framework of PE skills in educational research. Taking the inspiration from the Evidence-Centered Design (ECD; Mislevy and Haertel, 2007), we start by defining claims on how students are supposed to understand and use AI tools in the context of online reasoning. The insights from this paper will serve to inform the development of the ECD-based model of PE in future research.

Regarding PE, we make a distinction by the type of model it is applied to. In this paper, we focus on the LLMs, and not Text-to-Image Models, since they have their own specific manner of engineering prompts (Liu and Chilton, 2022; Oppenlaender, 2022). LLMs (or polymodal models) might be applied to a significantly wider variety of tasks, making them more flexible.

Moreover, we focus on the application of ChatGPT as the main and one of the most general AI-assisted tools. We also focus on the user side of PE, and not on the technical side of improving the model performance by specifically training it for the task. This machine learning subfield is also called PE, but it focuses on technicalities, like text embedding optimizations (Gu et al., 2023), or training on specific outputs indicating the nature of reasoning of a larger model (Mukherjee et al., 2023). Hence, for the sake of this paper, we exclude any procedure that implies re-estimation or optimization of the LMM parameters from the scope of PE and focus it exclusively on the user-side of LLM applications.

In addition, we make a distinction between PE as a (composite) skill and PE as a practice. PE as a practice has been described to some extent by other researchers (Cain, 2024; Wang et al., 2023b), and some showcase examples aimed at learning PE (Google, 2024). The description of PE practice is focused on unsystematized hints, tricks, and examples intended to help users achieve the desired result from an LLM. In the description of PE as a practice, many researchers emphasize that PE is often a continuous process that unfolds over several iterations of interactions between a human user and an LLM, much like many other information processing-related practices. This makes the description of the PE skill, like descriptions of many other information processing-related skills (Goldman and Brand-Gruwel, 2018) challenging because this structure needs to be able to incorporate the sequential aspect of the process. In such processes, many distinct cognitive components might activate in different orders or simultaneously, complicating their untangling for research investigations. However, the structure of the skills utilized in PE practice has been scarcely addressed, which serves as the motivation for this paper.

The (online) information literacy concept, regardless of the exact framework or definition, typically splits into passive (user) and active (developer) use (Koltay, 2011). In this paper, we discuss PE in the context of only passive use by (higher education) students for learning and knowledge acquisition, which corresponds to engineering prompts and evaluating LLM output. PE itself, however, can be considered under the frameworks of computer-assisted text production or creative writing, but these frameworks also lie beyond the focus of this paper as these aspects are more closely related to linguistics, media and communication science, rather than educational science.

While skill descriptions in Internet use and information acquisition might also apply to LLMs in general, our conceptual analysis illustrates that skills for competent use of LLMs (including PE) for learning differ from most skills in frameworks. Given the enormous interactivity of LLMs, their dependency on user input, their virtually unlimited knowledge, and their tendency to hallucinate while remaining very convincing, we conclude that PE requires distinct skills not covered by traditional (online) information literacy frameworks. To conceptualize a specific PE skillset, we review related prior skills frameworks, illustrating what we can learn from them, but also how they fall short in modeling specific PE (sub)skills, revealing a gap in the research.

4 Review of online skills frameworks and their distinctions from prompt engineering

When designing conceptual (and assessment) frameworks for 21st century skills related to AI, one first needs to identify the knowledge and skills students need while engaging with tasks. We, therefore, relate PE to a selection of prominent skill frameworks and show that they are too global, referring primarily to search-engine-based Internet inquiry, and do not cover LLMs, even lacking entire PE components (see also Zlatkin-Troitschanskaia et al., 2021). This section aims to illustrate that, although these skill frameworks are relevant for contextualizing PE in a pre-2023 Internet, they are currently insufficient for describing PE itself or in the context of the Internet in 2024.

4.1 Prompt engineering as part of (exploratory) technology use and targeted inquiry for information acquisition

Prompt Engineering (PE) for learning in (higher) education can be considered under at least two broader kinds of activities that relate to research on 'literacies': digital technology use and manipulation (as part of digital literacy) and information acquisition (as covered in information literacy) (for a differentiation, Koltay, 2011; for a synthesis for assessment of digital information literacy in higher education, Sparks et al., 2016).

In using digital technology, (higher) education students' understanding and use of LLMs (and corresponding tools like ChatGPT) can be examined as the ability to use a specific class of platforms for adequate purposes in ways to achieve desired results, e.g., to obtain textual output with specific qualities from an LLM. This paradigm highlights that the user

1. decides to consult an LLM (vs. other information resources) to acquire specific types of information,
2. interacts with the selected LLM,
3. ends the interaction when a compelling mental or emotional state is reached (e.g., satisfaction, frustration, tiredness, boredom), having either completed their inquiry or not.

Here, the motivation for selecting a specific LLM is important, and LLM(s) might or might not be the only source of information for the user. Regardless, PE in this context refers to the sub-phase of inputting and refining prompts and marks the user's main active input to the LLM to obtain desired information. This perspective helps frame students' general tool exploration and experimentation (among platform novices), their versatility in interacting through inputs, troubleshooting, ludic use, and unintended or original technology uses.

In the context of exploratory technology use, such as the inquiry of a new topic to assess a resource's usefulness, users may not necessarily aim for efficient prompting. Instead, they may seek to test the capabilities of an LLM within their domain of interest, evaluating factors such as breadth and depth of answers, as well as information quality. This testing may involve pushing the LLM to its limits to understand its full potential. By contrast, for specific inquiry, more skillful goal-directed users can be expected to seek to obtain only the types of information the present LLM can indeed produce (above their desired quality threshold). Thus, part of the PE skill set includes knowledge and understanding of what the LLM system can and cannot do to judge its suitability to a given task and use it only for as long as it is helpful (section 5.1). Advanced users can benefit from understanding the capabilities and limitations, ethics and privacy tradeoffs of different LLMs (including their multimodal data capabilities), the coverage of their training databases, reasoning capabilities, speed, energy use, cost, and other metadata. This knowledge enables them to select the most suitable set of tools for their inquiries.

4.2 Prompt engineering and online information problem-solving skills

In the tradition of research on information literacy, conceptualizations have combined general tool use for active production and (passive) interpretations of information, but have still been deficient in conceptualizing Internet-based skills (Foster and Piacentini, 2023). More applied conceptual approaches have sought to narrow this gap, from Multiple Document Literacy and Multiple Source Use/Comprehension/Understanding to Information Problem Solving on the Internet (IPS-I) (for an overview, Goldman and Brand-Gruwel, 2018; List and Alexander, 2019).

IPS-I (Brand-Gruwel et al., 2009) was derived as a descriptive model of (five) phases that users go through when solving tasks that require the acquisition of information expanded for the Internet. These phases cover analysis (problem analysis and prior knowledge activation, searching the Internet for information, preliminary and deeper processing of information along with evaluations and reflection) and synthesis (text response drafting, and process and product evaluation), accompanied by regulation (on task, time, test content). Despite the good description of the online inquiry phases in the IPS-I model, its explanation of sufficient reasons to conclude an online inquiry can be improved (Goldman and Brand-Gruwel, 2018). Moreover, IPS-I employs task covering typical online platforms but does not model their affordances explicitly, and does not yet include a differentiation of single-or multi-platform inquiry, or search engine vs. LLM querying.

In general, the IPS-I framework, although promising, needs to be adapted to LLMs. For instance, the search component is entirely geared toward search engines, and evaluation does not account for specific LLM cues or the pages-long machine-generated text (although it does highlight checking page ownership). Moreover, reasoning in IPS-I does not cover how the typical responses of LLMs can veil a lack of specificity or contain factually incorrect information.

The weighting of facets should differ for PE, as well. In evaluation, there are fewer website cues to be considered, while specific LLM language cues can become more important. Knowledge of LLM production becomes more crucial for discernment of output quality; for instance, domain knowledge which serves as a critical reference is typically still underdeveloped among learners. Regarding syntheses and reasoning, LLMs can effectively carry out part of the thinking for users. The key question becomes how satisfied, if at all, are users with the respective LLM output (when to cross-check with further sources or not), and what are possible pitfalls of LLMs to be hedged against. As LLMs can now forward queries to other LLMs and deliver results for several of the IPS-I inquiry subphases for the user, while augmenting the requirements for others such as prompt formulation, it remains to be seen in future descriptive studies if the IPS-I phase structure will hold for this new platform type.

4.3 Prompt engineering and PISA's framework 'learning in the digital world'

One of the flagships of the educational assessment practices is OECD's Programme for International Student Assessment (PISA). While it focuses on school students, PISA showcases assessment innovations and key 21st century skills targeted in each wave. With the advent of assessing broader skills, such as critical thinking and problem-solving in PISA (Pellegrino, 2023), the necessity of competent use of technology and fostering PE skills becomes increasingly evident. Assessing and unveiling educational needs to prepare students in leveraging technology effectively in an increasingly information-driven society, including the adept formulation of prompts for AI tools like ChatGPT, becomes a pivotal aspect in ensuring comprehensive educational outcomes.

The PISA 2025 model responds in part to recent technological trends with the Learning in the Digital World (LDW) framework. The authors break the skills of learning with and from software down into

(1) computational and scientific inquiry practices (analyzing problems and recognizing patterns, working with software outputs, conducting experiments and analyzing data),

(2) metacognitive monitoring and cognitive regulation (progress monitoring and adaptation, performance and knowledge evaluation), and.

(3) noncognitive regulation processes (maintaining task engagement and affective states).

While PISA's LDW framework is limited to offline administration to school students (thereby limiting its use for open search), a chatbot has been implemented, and exploratory tool use is designed intuitively so as to tap into (secondary school level) technology-based data manipulation and examination principles. On-task learning is included as part of the assessment and is measured through logged indicators constructed from situational inference rules. Inputs are limited to presented stimuli. However, the LDW framework (2025) also presupposes a closed information environment, which in this instance does not seem to include LLM-like tools (i.e., a highly versatile chatbot).

4.4 Prompt engineering and open (web) search assessments

The development of AI tools goes toe to toe with the assessment innovations, making the current educational assessment practices more authentic and complex (Sabatini et al., 2023). Assessment innovations have rapidly evolved in the recent past, also paving the way for more complex conceptualizations of skills (with the prospects of feasible operationalization). Particularly, there is a shift from closed information pools (including in multiple source use) to open (web) search assessment environments (Wineburg et al., 2022). This places much more emphasis on subskills such as targeted search, rigorous selection, and cross-referencing to obtain useful information, under the uncertainty connected to never seeing the entire information pool. This contrasts with the careful evaluation of every source and deduction of information from a limited information pool, as in assessments with a document library (Shavelson et al., 2019). Thus, open web search assessments, compared to closed ones, differentially tap and weight-assessed subskills in a more ecologically valid setup. They place importance on design features such as abundance vs. scarcity of acceptable quality alternatives, noise vs. no noise, access, familiarity, and affordances for information search and organization. However, they also raise the need to account for cheating opportunities and changes in the information pool.

LLMs can further increase these differences, while also synthesizing and filtering out some of the interim complexity of open web search assessments. LLMs are generative models, meaning that they can generate new texts that never existed before. Hence, many of the open web search characteristics also apply to their use. An LLM's capability to recombine, structure, preselect, and synthesize information (while leaving the undisplayed bits and even sources opaque) differentiates it from search engines and results in a corresponding weighting of required skills. Users need to put less thinking effort into drawing inferences and compiling an (initial) draft that summarizes their inquiry, as the system can do this step for them. Corrections are also facilitated. Compared to search engines, LLMs are more dialogical, e.g., in explicitly restating, reaffirming, specifying, correcting user's prompts (seemingly in their own terms), and are able to autonomously apply suggested changes to entire text blocks of results. Therefore, studying and assessing PE as a new education-relevant skill requires the application of sufficiently complex forms of assessment practices.

4.5 Prompt engineering and critical online reasoning skills

Critical Online Reasoning (COR) is a recent conceptualization of the skillset necessary to acquire, evaluate, and reason with and about sources and information from the Internet, developed for the setting of learning in higher education (Molerov et al, 2020; Nagel et al., 2020, 2022). COR provides a convenient conceptual adaptation and development of IPS-I phases to the process of solving complex, open-ended information problems in a mixed information quality environment (Molerov et al., 2020). In particular, COR includes three interconnecting facets: "Online Information Acquisition" (OIA), "Critical Information Evaluation" (CIE), and "Reasoning Using Evidence, Argumentation, and Synthesis" (REAS), as well as a meta-cognitive facet (MCA) for the situation-specific activation and regulation of the COR skills.

COR was developed to implement advances in assessment, specifically to include ecologically valid open web search in assessment (with associated web behavior tracking). COR has aspired to capture competent behaviors regarding differences in the credibility of online sources and information, focusing on students' discernment of Internet sources and content in the face of realistic challenges online, such as a multitude of information, low-quality information, and/or misinformation (Molerov et al., 2020).

COR, too, was conceptualized before the popularization of LLMs. Today, students may find satisfying answers on ChatGPT and avoid further search or synthesis. LLMs can offer recommendations on general evaluation criteria. By loading or copying texts and sources into the LLM as part of a prompt, students may also obtain full machine evaluations.

4.6 Prompt engineering and artificial intelligence literacy

One of the more recent concepts, Artificial Intelligence (AI) literacy, is also relevant to the discussion of PE. In response to the rapidly developing AI field, the educational science community has already begun attempting to define the skills needed for competent AI use. AI literacy concerns AI in general, which is far broader than the topic of using LLMs. The concept of AI literacy distinguishes between generic and domain-specific use of AI (Knoth et al., 2024a), which includes numerous AI tools and machine learning models developed for narrow use in specific professional fields. Moreover, the explicit inclusion of attitudes in AI literacy (Wang et al., 2023a) binds at least part of the operational indicators of the construct to the self-report format, which becomes troublesome in the case of developing educational assessments.

Attempts to include more objective measures of AI literacy, however, tend to focus on the general knowledge of respondents about the structure, nature, and functioning of AI (Hornberger et al., 2023; Weber et al., 2023), which is not the same as defining *what allows a person to use AI successfully,* albeit complementary to this ability. Moreover, the employed items have been of multiple-choice format, which increases standardization but can threaten the authenticity of the assessment and ecological validity of claims about the respondents.

Given that our definition of PE aims to isolate the cognitive nature of the skill specifically tailored for the use of LLMs, we conclude that this is a distinct skill, which might be considered a part of AI literacy but is not defined by it. Within AI literacy (i.e., among all available types of AIs), LLMs and their use comprise a portion of a specific type of AI. Within LLM use, PE comprises a significant portion of the skills needed. Moreover, AI literacy frameworks are still in the early stages and evolving, requiring more specificity in various subareas. While general considerations about the functioning of AIs apply to PE as well, most parts are still underspecified.

In summary, the abovementioned skill frameworks focus on the perspective of students as agentic learners who actively regulate their own learning processes. Therefore, PE and LLM use are compatible with the above frameworks, as they essentially relate to the consistently emerging cognitive, metacognitive, and self-regulatory skills (Foster, 2023a; Roll and Barhak-Rabinowitz, 2023). They can be attributed to the respective search phases as well as partly to the evaluation and reasoning phases, if the assessment's operationalization grants students access to LLMs. However, the frameworks are not specific enough in addressing how users can skillfully interact with the AI-supported tools like ChatGPT to obtain desired information, leaving a number of conceptual questions open. As with any new technology, novel affordances call for new (sub)skills, too.

5 Conceptualization of prompt engineering skills

For the sake of this paper, we define PE as the skill of communicating the problem, its context, and the constraints imposed on the desirable solution to an LLM to solve it correctly as fast as possible (Lo, 2023a). Thus, this skill conveys the user's needs to an LLM in a manner that the model can understand. However, since the tasks to which the LLMs can be applied vary, the prompts for their application vary as well (White et al., 2023), so there is no general "best" prompt structure. Instead, it makes sense to describe the PE as a composite skill consisting of a combination of several subskills involved in the communication with the LLM. We describe PE as a composite multidimensional skill consisting of four skills, intertwining in the practice of using an LLM (Figure 1).

Since the purpose of this paper is to develop an operationalization of PE that can be flexibly used in the development of various assessments, either in parts or entirely, we do not provide a taxonomy of behavioral indicators (e.g., Bloom's taxonomy; Anderson and Krathwohl, 2000). Such taxonomies are closely linked to the exact type of claims about respondents that the assessment aims to make and are therefore defined by the purpose of the assessment (Mislevy and Haertel, 2007). Describing any taxonomy of behavioral indicators in the context of PE would reduce the possible scope of such applications and purposes. Instead, we aim to broadly describe the structure of the PE construct, which can be developed and used further. The application of an incorrect taxonomy can result in the misspecification of the assessment framework, a decrease in the authenticity of the assessment, and a degradation of the validity of the final claims. To further enrich the understanding of PE, we juxtapose online skills frameworks to integrate and adapt their essence to PE skills.

5.1 Understanding the basic prompt structure

Giray (2023, p. 2630), following the DAIR.AI (2023) frameworks, lists four elements of a prompt. These elements, combined together, substitute a prompt, which formulates the problem, gives the model the necessary information to solve it, and contains output in the desired form:

- Instruction – a specific task that guides the model's behavior (e.g., "Proofread the text");
- Context – external information or additional context that provides background knowledge to the model, helping it generate relevant responses (e.g., "The text is an email that needs to follow an official corporate style");
- Input data – the content of the prompt that the model needs to solve, might vary given the instruction (e.g., contents of different emails);
- Output indicator – specifies the type or format of the desired output (e.g., "Do not rewrite the text, only correct grammar, spelling, and punctuation").

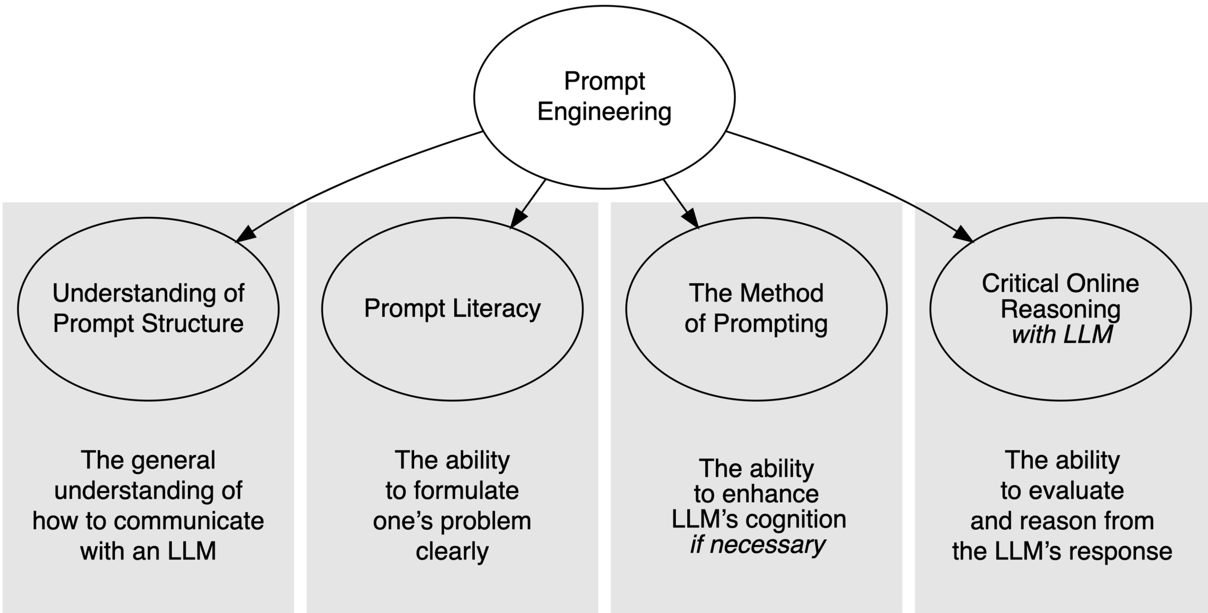

FIGURE 1
The proposed structure of PE.

Incorporating "output indicators" in the prompt structure implies that a user needs to have a projected image of the result. This means that PE inherently requires a user to know what they want. This will allow an AI tool to shape the output according to the user's expectations. This allows the user to have representational benchmarks, against which the output of an AI tool is judged.

In sum, understanding the necessity to provide all these elements, as well as the ability to optimize them when needed, can be considered the necessary part of PE. The quality of these components is particularly important, since imprecise formulations or irrelevant information can derail the LLM's response to the prompt.

5.2 Prompt literacy

Prompt literacy addresses the user's ability to be precise in their formulations. None of the research to date has to come up with any exhaustive lists of requirements for being precise in prompts. Still, Hwang et al. (2023) define prompt literacy as the ability to generate precise prompts as input for AI tools, interpret the outputs, and iteratively refine prompts to achieve desired results. Others vaguely address this literacy in terms of avoidance of pitfalls and common mistakes that learners make while engineering prompts (Busch et al., 2023; Lo, 2023b). Nonetheless, in practice, avoidance of the aforementioned pitfalls might not be necessary, as the improvement after an initial input or the general awareness of the limitations of one's suboptimal prompts allows for the avoidance of incorrect conclusions. Mostly, researchers highlight such aspects of prompt literacy as (Giray, 2023):

- Ambiguity or lack of specificity – without a concrete focused input, an LLM might wander away from a desired solution.
- Bias reinforcement – an ill-formulated prompt might provoke an LLM to give an answer which can be interpreted as biased.
- Overfitting and unrealistic dependency on model limitations – an LLM might not know all the specific details of a certain field or area, and as a result might be not the best consultant on an overly specific topic.
- The correct context – LLM needs *necessary and sufficient* context to work with (e.g., "write an email" is not a specific enough prompt to solve a problem correctly).
- Overly complex prompts – supplying too much information might trigger the LLM to focus on an irrelevant part of the prompt.
- Ethical considerations – ethical and ecological use of an AI that does not inherently have values or an ethical system remains the responsibility of the user.

The aspect of ethical considerations has received significant attention in PE literature. It relates to the fact that many LLMs have pre-built system prompts (hidden from the user) that explicitly prohibit them from discussing unethical topics (e.g., crimes or violence). Because of this, so-called "jailbreaking" has gained special attention in PE literature as a way to relax this limitation (Zhou et al., 2024; Yu et al., 2024). This topic is closely related to the general ethics of AI usage and is a specific field of AI research in general (Jobin et al., 2019) and AI research in education specifically (Borenstein and Howard, 2021; Burstein (2024) for the Duolingo Standards on Responsible AI). While a rigorous discussion of this topic is beyond the scope of this paper, the ethics of PE, as well as the ethics of a major part of general AI use, boils down to keeping a human in the loop of the process involving an AI tool and holding the person accountable for the decisions made (Shah, 2024).

Interestingly, in some cases, the context of the problem needs to be reduced rather than explicated. For instance, in the study by

10.3389/feduc.2024.1366434

Krupp et al. (2023), a physics problem was phrased along the following lines: "Tarzan swings hanging on a liana of a given length with a given speed from a given height. He picks up Jane (who has a given mass) standing still on the ground. Calculate Tarzan's speed right after he has picked Jane up." This phrasing can derail ChatGPT into discussing the Tarzan story, rendering the output useless. However, rephrasing the problem in terms of pendulums and loads will result in ChatGPT giving a correct response (or, in analogous cases, at least providing the user with the correct formulas for further calculations). This example illustrates that LLMs, like humans, can struggle to discern details from the core of the problem if the context is too unexpected. This behavior makes LLMs similar to humans who might experience the same difficulties (Carnoy et al., 2015). Therefore, a proficient level of PE requires the user to carefully measure the necessary and sufficient context for solving the given problem, not only expanding it (as is typically highlighted in the literature) but also reducing it in some cases.

In general, there is a striking similarity to item writing principles from test development and prompt literacy. For example, Haladyna and Rodriguez (2013) highlight 23 features that test items should have, among which, for example, are:

- Test important content, avoid overly specific and overly general content;
- Avoid opinions unless qualified;
- Avoid trick items;
- Edit and proof items;
- Keep linguistic complexity appropriate to the desired output;
- State the central idea clearly and concisely.

These similarities make sense conceptually, since, in both cases, the prompt/item writer is trying to be as precise, unambiguous, and economical as possible to achieve the purpose of prompt or assessment. This is attributed to the fact that, in both situations, a higher number of brief items can yield more reliable data, as the information can be amassed across a greater number of instances (Piacentini et al., 2023).

5.3 The method of prompting

The method of prompting is an aspect of PE that up until now has almost exclusively been studied from the technical perspective, or just anecdotally described by the users. The method of prompting is an inherent component of PE that includes using special verbal ways of organizing the prompt information to help an AI tool solve the posed problem. Although not all methods of prompting are suitable for every problem, it is crucial for users to understand these methods and identify when they are applicable. This knowledge can significantly improve the performance of LLMs.

In the following, we provide a relatively detailed discussion of the methods of prompting, as it offers practice-related insights into the functioning of LLMs. We suggest that understanding these aspects might be more important than the technically oriented knowledge of internal AI machinery, which tends to attract researchers' attention when they attempt to measure AI literacy. Hence, these methods of prompting might serve as a basis for the assessment of PE.

In general, there are many different methods of prompting. For example, Sahoo et al. (2024) describe 29 distinct techniques, most of which can be considered separate methods of prompting. However, some of them can be seen as variants of each other, and others are prompting strategies individually tailored to specific tasks. Moreover, new methods of prompting that require increasingly more skills than information processing (e.g., collective prompting that includes communication between human users and thus requires social communication skills; Wang et al., 2024) are continuously being created, making it impossible to exhaustively describe and systematize all recent prompting methods. Therefore, we limit ourselves to a brief description of some of the most prominent and important "families" of prompting methods.

5.3.1 Few-shots prompting

Few-Shots (FS) prompting[1] (Brown et al., 2020) may be important when an LLM is required to reason by analogy. FS prompting refers to the idea of providing the model with several examples of similar tasks and their solutions before the actual task. This idea bears a heavy similarity to Bandura and Jeffrey's (1973) observational learning concept, suggesting that people might learn something from observing other people doing it. The mechanics of LLMs' reasoning in this approach to prompting (the input-label mapping, the distribution of the input, the label space, and the output format; Min et al., 2022) details the "motor reproduction" process – one of four processes that account for learning according to Bandura and Jeffrey (1973), except, it unfolds in the verbal space.

Another process, retention, also has a reflection in the prompting literature since if the total length of the prompt exceeds the context memory of an LLM (the number of input tokens that the model uses to condition its responses), LLM's performance decreases (Mosbach et al., 2023; Kuratov et al., 2024). This exact phenomenon is fundamental to some of the "jailbreaking" techniques, which aim to overwhelm the context memory of an LLM to make it "forget" the ethical constraints contained in the latent system prompt (Jiang et al., 2024). However, with the recent chase after an exponential increase in the number of context tokens that an LLM can remember (up to millions of tokens; Reid et al., 2024; Zhang et al., 2024), these "jailbreaking" techniques become obsolete, and the problem of insufficient LLM context memory decreases. This chase unlocks other features of LLM use, such as the so-called mega-prompts (a couple of pages long) and the use of dozens of examples for few-shot (FS) learning.

Other processes from Bandura and Jeffrey (1973), attention and motivation, are barely covered in the prompting literature. However, while motivation is non-existent in AI literature in general on account of LLMs lacking it, the attention mechanism is almost solely responsible for LLMs' existence (Vaswani et al., 2017). This mechanism allows LLMs to find the dependencies between language tokens from different parts of a token sequence in a computationally efficient

1 Originally, this method of prompting has been termed *Few-shots (AI) learning* (Brown et al., 2020), and later the term has become *In-context (AI) learning* (Dong et al., 2022) but since we refer to AI learning as to the process of optimizing the model parameters, we have re-labelled this method of prompting to better reflect its nature.

manner. Still, this mechanism is an architectural feature and has no impact on prompting strategy.

FS prompting also reflects a somewhat traditional insight from psychological research in intelligence (Gentner et al., 2001) and higher-order reasoning (Alexander et al., 2016) which states that analogy is the fundamental concept of these processes. In the context of AI, an LLM having few examples of what is required from it can focus on the key aspects of the task better, generalize from them, and repeat the required information processing on the actual task.

FS prompting has had such an immense impact on the model performance that it has become a general practice in evaluating the model performance to reflect in the reporting documents what method of prompting exactly (e.g., 5-shots or zero-shots) has been used when several competing LLMs are measured against benchmark tasks (Bragg et al., 2021).

5.3.2 Chain-of-thought prompting

When an AI tool, for example, is required to perform some complex informational tasks (e.g., to formulate the implications of a text, or to reason from it in regards to a specific context), it needs to be allowed to spell out its reasoning steps (Kojima et al., 2022). Since LLMs do not have implicit higher-order reasoning skills (as they only *predict the next language token*), their reasoning can only occur in the form of "thinking aloud." This method of prompting in particular resembles concurrent thinking aloud (Fuchs et al., 2019). If the core of a request to an LLM includes several complex operations on the textual information, requiring the model to explicitly describe the steps that lead to its conclusion invokes the higher-order reasoning skills in the model and sufficiently improves the quality of the output (Wei et al., 2022). Such method of prompting is called *Chain-of-Thought* (CoT) prompting.

However, not all tasks require CoT prompting as, for example, some requests may just require creating a simple overview of a topic or rewriting a text. Hence, knowing about this method of prompting and recognizing when and how to have a model "think aloud" is also required from a user. This prompting method also has implications for the machine learning community, since training the model on the datasets that explicitly describe those reasoning steps can sufficiently boost its intelligence, even if the size of the model is relatively small. In such cases, the model trains to mimic the reasoning steps described in the training corpus, which is sufficient to exhibit impressive reasoning skills in the model evaluation (Mukherjee et al., 2023; Mitra et al., 2023).

Currently, CoT prompting has sparked a separate area of research in PE (Sahoo et al., 2024). For example, CoT promoting has been generalized to Graph-of-Thought (Yao Y. et al., 2023) and X-of-Thought (Ding et al., 2023) reasoning strategies that force LLMs to learn to reason internally, without spelling the solution process out. A significant portion of such research is dedicated to "interiorizing" this higher-order reasoning in LLM. The purpose of this "interiorizing" is essentially to make LLMs automatically (in a hidden manner, "internally") apply the reasoning steps without spelling the reasoning steps out ("externally"). This appears to be the key to unlocking the extremely complex cognitive performance of AI (Chu et al., 2023). This research direction bears similarities to Vygotsky's concept of interiorization, which states that higher psychological functions initially develop with external support in the real world and then become executed internally within the human mind without requiring this support (Bertau and Karsten, 2018). Importantly, these similarities are only superficial, since Vygotsky described the development of human psychological phenomena.

5.3.3 Tree-of-thought prompting

While this is a generalization of CoT (Yao S. et al., 2023; Yao Y. et al., 2023; Long, 2023), it has gained particular prominence. While some technical implementations of ToT require coding applications, a non-technical prompting variant has been suggested. It requires a prompt that emulates a collaborative brainstorming session among experts (Al-Samarraie and Hurmuzan, 2018). Hulbert (2023) uses the following prompt: "*Imagine three different experts are answering this question. All experts will write down 1 step in their thinking, then share it with the group. Then all experts will move to the next step,* etc. *If any expert realizes at any point that they are wrong, then they leave. The question is...*" This method enables the model to fulfill multiple roles and potentially enhances its performance.

Self-Consistency (Wang et al., 2022) is another technique used to enhance model performance. Essentially, this method poses the same query to the model multiple times and determines the most frequently occurring response. While there are various sophisticated methods to refine this procedure (Wang et al., 2022), it is also possible to apply it in a straightforward, manual fashion, ensuring that the model does not retain memory of previous responses (e.g., by initiating new chat sessions). This concept is akin to the *wisdom of crowds* (Surowiecki, 2005), which posits that a collective group of individuals often makes more accurate judgments than individual members of the group. In the context of LLMs, the model is treated as if it were a crowd of people, with each new attempt at answering the question acting as an independent opinion from the group. It is crucial, however, for the user to ensure that each response remains separate from the previous one – repeating the question in a continuous chat thread may lead to biased reasoning due to influence from the previous attempts.

Self-fact-checking is another strategy (Semnani et al., 2023), which helps to mitigate LLM hallucinations. While originally designed as a chat-bot function, users can manually adopt this technique. Here, the response is divided into individual claims, verifying their accuracy separately, and constructing a final response only from those which are correct. Although the self-fact-checking chat-bot has shown superior performance compared to other LLMs, it operates more slowly due to the additional steps involved. Nevertheless, users can incorporate elements of this method by inquiring about the veracity of sources or specific facts. This practice draws strong parallels with retrospective thinking aloud (Prokop et al., 2020), focusing on the evaluation of information rather than its generation.

5.3.4 Role-model

Another strategy relating to the method of prompting are special role-model hints that a user can have an LLM consider when answering the request. Such approaches have been described only anecdotally to date (Ivanovs, 2023). For example, some users have noticed that ChatGPT can perform better if it has been offered money for the successful solution.[2] Although this is an obviously nonsensical statement, adding this suggestion to the prompt

2 https://x.com/voooooogel/status/1730726749854663093?s=20

evidently increases the meticulousness of ChatGPT's response. It has been suggested that this is an artifact of the dataset that was used for the ChatGPT training. Since it included some Internet forums where users ask for help, some of such requests included the promise of monetary prizes for those that would help to overcome the problem. Correspondingly, the solutions provided to such requests were more verbally rich, rigorous and meticulous, and better overall. Hence, once tokens meaning the promise of money for the solution (or similar ones—for example, saying that the user's work or life depends on the success of the solution) are used in the prompt, ChatGPT imitates the responses that were given to similar requests on the Internet. This is consistent with the intuition of neural networks learning the data features, which can be overlooked by the creators of the training datasets but are still present (Buolamwini, 2017). In terms of superficial psychological analogies, this calls for observational learning (Greer et al., 2007) to be externally motivated (Hendijani et al., 2016).

The list of similar role-model hints is constantly increasing, as users discover new saddle features of ChatGPT's behavior. Some recommendations to date include:

- Asking LLM to "take a deep breath" (because, apparently, this combination of tokens is used when people describe the successful solutions of the problem after a long and frustrating chain of attempts; Yang et al., 2023),
- Asking ChatGPT to imagine, that it is now May (that is related to the fact that ChatGPT also receives a latent timestamp of the prompt as well as the training dataset also having timestamps; apparently, close to holidays (especially in December), the length of human responses which were contained in the training dataset decreased, resulting in ChatGPT giving more concise responses leading up to and after wide-spread holidays[3]),
- Stating that a user "unfortunately has no fingers," so "they cannot type" (apparently, it is especially successful in the request of writing programming code; it makes ChatGPT provide a final solution to the problem, incorporating all small changes to the final code at the same time; Ivanovs, 2023).
- Additionally, several other tricks, such as making the LLM repeat the question before answering or stressing human-relevant motivation factors (Bsharat et al., 2023), appear to have a positive impact on LLM performance.

5.4 Toward specifying prompt engineering in relation to critical online reasoning

Walter (2024) has suggested that the ability to critically evaluate the output of an LLM is a crucial part of successfully integrating AI into educational processes, alongside prompting skills. Additionally, Krupp et al. (2023) found that one of the major problems in students' use of LLMs is the lack of critical evaluation of LLM outputs. Given that the prompting structure (section 5.1) includes output indicators, PE implicitly requires the user to conceptualize the desired LLM output and evaluate the actual output against it. We suggest that this set of skills is necessary at the stage of evaluating LLM output and deciding on further actions.

With regard to the COR facets (section 4.5), particularly the evaluation (CIE) and reasoning (REAS) facets are necessary for concluding whether or not the LLM output satisfies the necessary criteria of the desired solution. The information acquisition facet (OIA) is relevant at the stages of selecting a platform such as an LLM for a (sub) inquiry, choosing between several available LLMs, and formulating prompts. When formulating a relevant prompt, the user is responsible for correctly phrasing and articulating the prompt in relation to the actual problem they are trying to solve.

The meta-cognitive facet (MCA) is related to the motivation of the user to critically evaluate the LLM output. This facet is the most elusive in the COR structure since it is hard to operationally disentangle *low motivation to use COR abilities* from *low COR abilities*. This problem is one of the most important in the field of assessment of higher-order cognitive skills in general and 21st century skills in particular, as they by definition include this meta-cognitive component. Current assessments presume motivation and awareness as given within the test-taking window, thanks to extrinsic motivators (i.e., test-taking incentives) and explicit task instructions.

The features making COR an important component of PE include its interactivity, high emphasis on ecological validity, focus on information quality and web behavior tracking capabilities. Given that LLMs are by definition interactive, if a user finds the output of an LLM unsatisfactory, they might change the prompt on the spot or correct it in natural language immediately after evaluating the output. This significantly alters traditional understandings and conceptualizations of critical reasoning because they are often defined and measured in much less interactive environments.[4]

Moreover, high authenticity and ecological validity have been crucial parts of the COR skills from the very beginning of their development, demanding innovative assessment formats that do not restrict the natural unfolding of these processes by utilizing traditional standardized and well-studied response formats (such as multiple choice). Additional features of COR skills, such as their connection to the online environment and their utility for filtering out fake information (Molerov et al., 2020), strengthen the tie of this skillset to PE. Given LLMs' propensity to hallucinate and invent non-existent information (such as imaginary sources), the focus in COR skills on source quality evaluation is highly relevant.

6 Implications and challenges for developing a prompt engineering assessment framework

The purpose of this paper is not to develop an assessment framework of PE but to provide an initial conceptualization of the PE construct as a skill set that enables a person to use LLMs successfully. This has implications for the corresponding assessment framework to be developed in the future. Section 5 might serve as a PE construct

3 https://x.com/RobLynch99/status/1734278713762549970?s=20

4 For an exception, see Jahn and Kenner (2018), whose 4 phases model synthesizes critical thinking into both a receptive and an interactive half arch; the latter includes hypothesis formation and testing.

model within the terminology of ECD (Mislevy and Haertel, 2007), as it lists the proposed conceptual components of PE and broadly describes their content. However, not all components might be used in an assessment if one is willing to accept the limitations of the claims about the respondents that a selective construct model may entail. Moreover, some components of the constructed model can be added to the proposed structure if this is justified for a given assessment. For example, it is expected that different LLMs perform better when solving different kinds of problems. Hence, the general awareness of specific proficiencies of different LLMs or ethical considerations in LLM use can be designated as separate components of PE if necessary.

We intentionally refrain, however, from providing an ECD evidence model in this paper. Given that PE is a 21st century skill, its assessments can be developed for an enormously wide range of situations. From formative assessments in high schools to high-stakes assessments in recruiting, different aspects of PE might be more or less relevant for different contexts. Given the inherent connection of the evidence model with the assessment context, we do not provide any limitations on the types of evidence that can be used to assess PE as a skill.

However, when it comes to the ECD task model, one issue becomes abundantly clear: it is nearly impossible to assess PE with the traditional multiple-choice response format. The inherent property of PE—interactivity—demands innovative response formats for any PE assessment. The necessity for a highly authentic task model, in turn, impacts the scoring procedure. While psychometricians are understandably comfortable with traditional response formats, utilizing them for the assessment of such highly complex skills appears to be a misspecification of the assessment framework.

The use of restricted virtual chats similar to PISA's collaborative problem-solving assessment (OECD, 2017) is a possible approach here. In such assessments, the multiple-choice items are masked by students selecting pre-formulated replies suggested by a test developer, along with a coherent storyline and rescue points mimicking learning progression toward the relevant outcomes (Piacentini et al., 2023). However, while the limited variability and flexibility of suggested responses in such virtual chats can be advantageous for assessment standardization, the assessment's authenticity is still decreased in this option. Creating choice-rich environments is a very complex task on its own (Piacentini et al., 2023).

Instead, the use of open-ended response formats is appealing for PE assessment. They appear to be highly effective in capturing the shifting assessment purpose from summatively evaluating the presence of static knowledge to evaluating students' ability to acquire and scrutinize knowledge in different contexts (Roll and Barhak-Rabinowitz, 2023). In the face of this paradigm shift, it has been asserted that generating, assessing, and processing such complex data streams from these interactive tasks is feasible on a large scale only through the utilization of advanced digital technologies (Hu et al., 2023). Additionally, more robust claims can be made on students' (differential) skills if they work on invention activities in an unconstrained (or less constrained) environment (Piacentini et al., 2023).

The scoring of open-ended items, however, is usually done with human raters, prohibiting interactivity and making such assessments very expensive. Here, the utilization of other language models – for automated scoring of open-ended responses (LaFlair et al., 2023), as well as other innovations from the field of automated LLM evaluation with specific evaluator language models (Kim et al., 2024) – can improve the economic feasibility of such assessments. Still, such scoring procedures will be based on uninterpretable statistical models scoring the responses, which can be considered a threat to validity (Lottridge et al., 2023).

Importantly, this also impacts procedural aspects of the assessment structure, such as the ECD delivery model. Until Small Language Models (Zhu et al., 2024) can be utilized as efficiently as LLMs, such technologically enhanced assessments will predictably require not only a computer but also a stable online connection. In general, as with any assessment, designing a PE assessment appears to involve a complex network of trade-offs between multiple aspects, with no solution fitting all situations.

Given all these implications, one of the most significant potential advantages of PE assessment is its possible orientation for learning, which goes much deeper than traditional formative assessment (Hu and Wang, 2024). The provision of learning resources through ChatGPT in assessment tasks can synergistically serve multiple purposes: enabling non-linear learning trajectories, facilitating interactivity for meaning-making, capturing digital traces to unveil intermittent cognitive processes, etc. Only tasks that trigger deeper learning can unveil misconceptions and faulty strategies, identifying further needs for support in follow-up training (Piacentini et al., 2023). This approach provides limitless opportunities for assessing self-regulated learning (Roll and Barhak-Rabinowitz, 2023). The interactivity and adaptability of LLMs can tailor challenges to different abilities, improving measurement quality and the authenticity of assessments (Piacentini et al., 2023) while maintaining student engagement in the assessment (Foster, 2023b). Overall, the thoughtful implementation of a PE assessment has the potential to grow into an unprecedented assessment-for-learning tool with capabilities previously unseen.

7 Conclusion

With the development of AI assisting tools, multiple areas of human learning are experiencing rapid changes. AI promises to revolutionize nearly all fields of information processing – from high-stakes decision making to education. This is especially evident in the discussions around AI-based chatbots like ChatGPT, which are based upon the use of LLMs – machine learning engines which create a sequence of language tokens (words, letters, and punctuation) as an output in response to an initial prompt from the user. Still, multiple reports have emerged stating that incorrect phrasing or an inappropriate context of the problem is capable of degrading the output of an LLM beyond any use. These reports highlight that the skill of communicating with an LLM—PE—might be as important as the AI-assisted tool itself. Moreover, since LLMs can be applied universally across nearly all areas of learning and professional activity, this skill should be conceptualized as a universal skill, similarly to the widely-recognized 21st century skills. Given that the rise of AI and its applications is expected to be increasingly wide-spread, the necessity for studying this skill in the field of educational science becomes evident (Gattupalli et al., 2023). This paper constitutes one of the first approaches to this topic, attempting to justify such investigation of this skill in the tradition of educational assessment.

We demonstrate that this emerging skillset is not covered by the existing frameworks for the 21st century skills, although, it fits within them nicely. Therefore, we suggest understanding PE as a composite skill reflecting people's ability to communicate with an LLM to solve informational problems and/or more complex disciplinary and cross-disciplinary tasks. This skill includes components reflecting the understanding of the basic prompt structure that is required for an LLM to understand the request, as well as the ability to navigate through the pitfalls of inappropriate formulation of the request. We show that the latter component, prompt literacy, bears a striking similarity to the item writing guidelines from the test development field, meaning that test developers already have a head start in understanding the art of PE. Moreover, PE requires an alternation between formulating the request and evaluating the output of the model to improve, reformulate, or stop and use the current solution provided by an LLM. This component is covered via critical online reasoning skills. Additionally, in prompting methods, we discuss different tricks in information organization and phrasing that can significantly increase LLMs' performance.

While we discuss some implications and challenges for AI assessment framework based on the initial conceptual framework of PE, concrete recommendations on the practices of the PE assessment or attempts to assess this construct lie far beyond the scope of this conceptual paper. Such a task, as well as the specific suggestion for the assessment framework of PT, would require numerous further theoretical and methodological investigations. This paper aims to contribute to the initial milestone of such a challenging endeavor and to justify the approach to the analysis of PE as a new 21st century skill, to outline its possible conceptual structure, and to call for further research.

We must critically recognize, however, that the current LLM development boom is outpacing many peer-reviewed academic research processes, assessment development cycles, and likely our abilities to maintain a sufficiently up-to-date overview. By the time a paper is completed, results and recommendations may become outdated and will certainly be incomplete. This paper does not attempt to be exhaustive; instead, it aims to initiate the discussion on PE as a skill relevant to professionals in the 21st century.

Moreover, the size of challenges in obtaining useful and credible content from LLMs keeps shifting as single facets of the inquiry process are augmented by new features. Thus, PE advice and skill components are bound to a specific time and system version (Chen et al., 2023b). They can become socially differentiated as experienced users may perform information quality or utility-enhancing services, such as offering ready prompts, prompt generation recommendations, or tools.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

DF: Conceptualization, Writing – original draft, Writing – review & editing. DM: Investigation, Writing – original draft, Writing – review & editing. OZ-T: Conceptualization, Funding acquisition, Investigation, Project administration, Supervision, Writing – original draft, Writing – review & editing. AM: Investigation, Writing – original draft, Writing – review & editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This research work was conducted in the context of the research Unit CORE (Critical Online Reasoning in Higher Education) funded, by the German Research Foundation (Funding number: 5404).

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Abbas, M., Jam, F. A., and Khan, T. I. (2024). Is it harmful or helpful? Examining the causes and consequences of generative AI usage among university students. *Int. J. Educ. Technol. High. Educ.* 21:10. doi: 10.1186/s41239-024-00444-7

Alexander, P. A., Singer, L. M., Jablansky, S., and Hattan, C. (2016). Relational reasoning in word and in figure. *J. Educ. Psychol.* 108, 1140–1152. doi: 10.1037/edu0000110

Alkaissi, H., and McFarlane, S. I. (2023). Artificial hallucinations in ChatGPT: implications in scientific writing. *Cureus* 15:e35179. doi: 10.7759/cureus.35179

Al-Samarraie, H., and Hurmuzan, S. (2018). A review of brainstorming techniques in higher education. *Think. Skills Creat.* 27, 78–91. doi: 10.1016/j.tsc.2017.12.002

Anderson, L. W., and Krathwohl, D. R. (2000). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. Upper Saddle River: Pearson Education (US).

Bandura, A., and Jeffrey, R. W. (1973). Role of symbolic coding and rehearsal processes in observational learning. *J. Pers. Soc. Psychol.* 26, 122–130. doi: 10.1037/h0034205

Bertau, M. C., and Karsten, A. (2018). Reconsidering interiorization: self moving across language spacetimes. *New Ideas Psychol.* 49, 7–17. doi: 10.1016/j.newideapsych.2017.12.001

Borenstein, J., and Howard, A. (2021). Emerging challenges in AI and the need for AI ethics education. *AI Ethics* 1, 61–65. doi: 10.1007/s43681-020-00002-7

Bragg, J., Cohan, A., Lo, K., and Beltagy, I. (2021). Flex: unifying evaluation for few-shot nlp. *arXiv*, 1–20. doi: 10.48550/arXiv.2107.07170

Brand-Gruwel, S., Wopereis, I., and Walraven, A. (2009). A descriptive model of information problem solving while using internet. *Comput. Educ.* 53, 1207–1217. doi: 10.1016/j.compedu.2009.06.004

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., et al. (2020). Language models are few-shot learners. *Adv. Neural Inf. Proces. Syst.* 33, 1877–1901. doi: 10.48550/arXiv.2005.14165

Bsharat, S. M., Myrzakhan, A., and Shen, Z. (2023). Principled instructions are all you need for questioning LLaMA-1/2, GPT-3.5/4. *arXiv*, 1–26. doi: 10.48550/arXiv.2312.16171

Buolamwini, J.A. (2017). Gender shades: Intersectional phenotypic and demographic evaluation of face datasets and gender classifiers. Master dissertation. Cambridge (MA), Massachusetts Institute of Technology

Burstein, J. (2024). Responsible AI standards. Duolingo. Available at: https://duolingo-papers.s3.amazonaws.com/other/DET+Responsible+AI+Standards+-+040824.pdf

Busch, K., Rochlitzer, A., Sola, D., and Leopold, H. (2023). "Just tell me: prompt engineering in business process management," in Enterprise, business-process and information systems modeling, eds. AaH. van der, D. Bork, H. A. Proper and R. Schmidt Cham: Springer Nature Switzerland, 3–11

Cain, W. (2024). Prompting change: exploring prompt engineering in large language model AI and its potential to transform education. *TechTrends* 68, 47–57. doi: 10.1007/s11528-023-00896-0

Carnoy, M., Khavenson, T., and Ivanova, A. (2015). Using TIMSS and PISA results to inform educational policy: a study of Russia and its neighbours. *J. Compar. Int. Educ.* 45, 248–271. doi: 10.1080/03057925.2013.855002

Chen, E., Huang, R., Chen, H. S., Tseng, Y. H., and Li, L. Y. (2023a). GPTutor: a ChatGPT-powered programming tool for code explanation. *arXiv*, 1–26. doi: 10.48550/arXiv.2305.01863

Chen, L., Zaharia, M., and Zou, J. (2023b). How is ChatGPT's behavior changing over time? *Harvard Data Science Rev.* 6, 1–47. doi: 10.1162/99608f92.5317da47

Chu, Z., Chen, J., Chen, Q., Yu, W., He, T., Wang, H., et al. (2023). A survey of chain of thought reasoning: advances, frontiers and future. *arXiv*, 1–31. doi: 10.48550/arXiv.2309.15402

DAIR.AI. (2023). Elements of a prompt. Available at: https://www.promptingguide.ai/introduction/elements (Accessed December 22, 2023)

Ding, R., Zhang, C., Wang, L., Xu, Y., Ma, M., Zhang, W., et al. (2023). Everything of thoughts: defying the law of penrose triangle for thought generation. *arXiv*, 1–34. doi: 10.48550/arXiv.2311.04254

Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., et al. (2022). A survey for in-context learning. *arXiv*, 1–22. doi: 10.48550/arXiv.2301.00234

Ekin, S. (2023). Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices. *TechRxiv*, 1–11. doi: 10.36227/techrxiv.22683919.v2

Foster, N. (2023a). "21st century competencies: challenges in education and assessment" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 30–44.

Foster, N. (2023b). "Exploiting technology to innovate assessment" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 98–109.

Foster, N., and Piacentini, M. (2023). Innovating assessments to measure and support complex skills. OECD Publishing. doi: 10.1787/e5f3e341-en

Fuchs, L. S., Äikäs, A., Björn, P. M., Kyttälä, M., and Hakkarainen, A. (2019). Accelerating mathematics word problem solving performance and efficacy with think-aloud strategies. *South Afr. J. Childhood Educ.* 9, 1–10. doi: 10.4102/sajce.v9i1.716

Gattupalli, S., Maloy, R.W., and Edwards, S.A. (2023). Prompt Literacy: A Pivotal Educational Skill in the Age of AI. *College of Education Working Papers and Reports Series 6*. University of Massachusetts Amherst. doi: 10.7275/3498-wx48

Gentner, D., Holyoak, K. J., and Kokinov, B. N. (2001). The analogical mind. Cambridge: The MIT Press.

Giray, L. (2023). Prompt engineering with ChatGPT: a guide for academic writers. *Ann. Biomed. Eng.* 51, 2629–2633. doi: 10.1007/s10439-023-03272-4

Goldman, S. R., and Brand-Gruwel, S. (2018). "Learning from multiple sources in a digital society" in International handbook of the learning sciences. eds. F. Fischer, C. E. Hmelo-Silver, S. R. Goldman and P. Reimann (New York: Routledge), 86–95.

Google. (2024). Prompting guide 101: A quick-start handbook for effective prompts. Available at: https://services.google.com/fh/files/misc/gemini-for-google-workspace-prompting-guide-101.pdf (Accessed December 22, 2023).

Greer, R. D., Dudek-Singer, J., and Gautreaux, G. (2007). "Observational learning" in Behavior analysis around the world: A special issue of the international journal of psychology. ed. C. Dalbert (London: Psychology Press), 486–499.

Gu, J., Han, Z., Chen, S., Beirami, A., He, B., Zhang, G., et al. (2023). A systematic survey of prompt engineering on vision-language foundation models. *arXiv*, 1–21. doi: 10.48550/arXiv.2307.12980

Haladyna, T. M., and Rodriguez, M. C. (2013). Developing and validating test items. New York: Routledge.

Hendijani, R., Bischak, D. P., Arvai, J., and Dugar, S. (2016). Intrinsic motivation, external reward, and their effect on overall motivation and performance. *Hum. Perform.* 29, 251–274. doi: 10.1080/08959285.2016.1157595

Hornberger, M., Bewersdorff, A., and Nerdel, C. (2023). What do university students know about artificial intelligence? Development and validation of an AI literacy test. *Comput. Educ.* 5:100165. doi: 10.1016/j.caeai.2023.100165

Hosseini, M., Gao, C. A., Liebovitz, D. M., Carvalho, A. M., Ahmad, F. S., Luo, Y., et al. (2023). An exploratory survey about using ChatGPT in education, healthcare, and research. *medRxiv*, 1–21. doi: 10.1101/2023.03.31.23287979

Hu, X., Shubeck, K., and Sabatini, J. (2023). "Artificial intelligence-enabled adaptive assessments with intelligent tutors" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 173–187.

Hu, S., and Wang, X. (2024). FOKE: a personalized and explainable education framework integrating foundation models, knowledge graphs, and prompt engineering. *arXiv*, 1–17. doi: 10.48550/arXiv.2405.03734

Hulbert, D. (2023). Using tree-of-thought prompting to boost ChatGPT's reasoning. Available at: https://medium.com/@dave1010/using-tree-of-thought-prompting-to-boost-chatgpts-reasoning-318914eb0e76 (Accessed January 4, 2024).

Hwang, Y., Lee, J. H., and Shin, D. (2023). What is prompt literacy? An exploratory study of language learners' development of new literacy skill using generative AI. *arXiv*. doi: 10.48550/arXiv.2311.05373

Ivanovs, A. (2023). Users are turning to reinforcement prompts to fix ChatGPT laziness. Available at: https://stackdiary.com/users-are-turning-to-reinforcement-prompts-to-fix-chatgpt-laziness/ (Accessed January 4, 2024)

Jahn, D., and Kenner, A. (2018). "Critical thinking in higher education: how to foster it using digital media" in The digital turn in higher education. eds. D. Kergel, B. Heidkamp, P. K. Telléus, T. Rachwal and S. Nowakowski (Wiesbaden: Springer Fachmedien Wiesbaden), 81–109.

Jiang, F., Xu, Z., Niu, L., Xiang, Z., Ramasubramanian, B., Li, B., et al. (2024). ArtPrompt: ASCII art-based jailbreak attacks against aligned LLMs. *arXiv*, 1–17. doi: 10.48550/arXiv.2402.11753

Jobin, A., Ienca, M., and Vayena, E. (2019). The global landscape of AI ethics guidelines. *Nat Machine Intelligence* 1, 389–399. doi: 10.1038/s42256-019-0088-2

Kaarakainen, M. T., Kivinen, O., and Vainio, T. (2018). Performance-based testing for ICT skills assessing: a case study of students and teachers' ICT skills in Finnish schools. *Univ. Access Inf. Soc.* 17, 349–360. doi: 10.1007/s10209-017-0553-9

Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., et al. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learn. Individ. Differ.* 103:102274. doi: 10.1016/j.lindif.2023.102274

Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., et al. (2024). Prometheus 2: an open source language model specialized in evaluating other language models. *arXiv*, 1–16. doi: 10.48550/arXiv.2405.01535

Knoth, N., Decker, M., Laupichler, M. C., Pinski, M., Buchholtz, N., Bata, K., et al. (2024a). Developing a holistic AI literacy assessment matrix–bridging generic, domain-specific, and ethical competencies. *Comput. Educ. Open* 6:100177. doi: 10.1016/j.caeo.2024.100177

Knoth, N., Tolzin, A., Janson, A., and Leimeister, J. M. (2024b). AI literacy and its implications for prompt engineering strategies. *Comput. Educ.* 6:100225. doi: 10.1016/j.caeai.2024.100225

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *arXiv*, 1–42. doi: 10.48550/arXiv.2205.11916

Koltay, T. (2011). The media and the literacies: media literacy, information literacy, digital literacy. *Media Cult. Soc.* 33, 211–221. doi: 10.1177/0163443710393382

Krupp, L., Steinert, S., Kiefer-Emmanouilidis, M., Avila, K. E., Lukowicz, P., Kuhn, J., et al. (2023). Unreflected acceptance--investigating the negative consequences of ChatGPT-assisted problem solving in physics education. *arXiv*, 1–9. doi: 10.48550/arXiv.2309.03087

Kuratov, Y., Bulatov, A., Anokhin, P., Sorokin, D., Sorokin, A., and Burtsev, M. (2024). In Search of needles in a 10M haystack: Recurrent memory finds what LLMs Miss. *arXiv*. doi: 10.48550/arXiv.2402.10790

LaFlair, G., Yancey, K., Settles, B., and von Davier, A. A. (2023). "Computational psychometrics for digital-first assessments: a blend of ML and psychometrics for item generation and scoring", eds. V. Yaneva and M. von Davier *Advancing natural language processing in educational assessment*. (Routledge), 107–123.

List, A., and Alexander, P. A. (2019). Toward an integrated framework of multiple text use. *Educ. Psychol.* 54, 20–39. doi: 10.1080/00461520.2018.1505514

Liu, V., and Chilton, L. B. (2022). Design guidelines for prompt engineering text-to-image generative models. *arXiv*, 1–26. doi: 10.48550/arXiv.2109.06977

Lo, L. S. (2023a). The art and science of prompt engineering: a new literacy in the information age. *Internet Ref. Serv. Q.* 27, 203–210. doi: 10.1080/10875301.2023.2227621

Lo, L. S. (2023b). The CLEAR path: a framework for enhancing information literacy through prompt engineering. *J. Acad. Librariansh.* 49:102720. doi: 10.1016/j.acalib.2023.102720

Long, J. (2023). Large language model guided tree-of-thought. *arXiv*, 1–11. doi: 10.48550/arXiv.2305.08291

Lottridge, S., Ormerod, C., and Jafari, A. (2023). "Psychometric considerations when using deep learning for automated scoring", eds. V. Yaneva and M. von Davier *Advancing natural language processing in educational assessment*. (Routledge), 15–30.

Maharajan, J., Garikipati, A., Singh, N. P., Cyrus, L., Sharma, M., Ciobanu, M., et al. (2024). OpenMedLM: Prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models. *Scientific Reports*. 14:14156. doi: 10.1038/s41598-024-64827-6

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., et al. (2022). Rethinking the role of demonstrations: what makes in-context learning work? *arXiv*, 1–9. doi: 10.48550/arXiv.2202.12837

Mislevy, R., and Haertel, G. (2007). Implications of evidence-centered design for educational testing. *Educ. Meas. Issues Pract.* 25, 6–20. doi: 10.1111/j.1745-3992.2006.00075.x

Mitra, A., Del Corro, L., Mahajan, S., Codas, A., Simoes, C., Agarwal, S., et al. (2023). Orca 2: teaching small language models how to reason. *arXiv*, 1–53. doi: 10.48550/arXiv.2311.11045

Mohr, G., Reinmann, G., Blüthmann, N., Lübcke, E., and Kreinsen, M. (2023). Übersicht zu Chat-GPT im Kontext Hochschullehre. Hamburg: Hamburger Zentrum für Universitäres Lehren und Lernen. Hamburg University.

Molerov, D., Zlatkin-Troitschanskaia, O., Nagel, M.-T., Brückner, S., Schmidt, S., and Shavelson, R. J. (2020). Assessing University Students' Critical Online Reasoning Ability: A Conceptual and Assessment Framework With Preliminary Evidence. 5:1102. doi: 10.3389/feduc.2020.577843

Morton, J. (2024). Using prompt engineering to better communicate with people. Available at: https://hbr.org/2024/01/using-prompt-engineering-to-better-communicate-with-people (Accessed February 15, 2024)

Mosbach, M., Pimentel, T., Ravfogel, S., Klakow, D., and Elazar, Y. (2023). Few-shot fine-tuning vs. in-context learning: a fair comparison and evaluation. *arXiv*, 1–29. doi: 10.48550/arXiv.2305.16938

Mukherjee, S., Mitra, A., Jawahar, G., Agarwal, S., Palangi, H., and Awadallah, A. (2023). Orca: progressive learning from complex explanation traces of gpt-4. *arXiv*, 1–51. doi: 10.48550/arXiv.2306.02707

Nagel, M.-T., Schäfer, S., Zlatkin-Troitschanskaia, O., Schemer, C., Maurer, M., Molerov, D., et al. (2020). How Do University Students' Web Search Behavior, Website Characteristics, and the Interaction of Both Influence Students' Critical Online Reasoning? *Frontiers in Education*, 5:565062. doi: 10.3389/feduc.2020.565062

Nagel, M.-T., Zlatkin-Troitschanskaia, O., and Molerov, D. (2022). Validation of newly developed tasks for the assessment of generic Critical Online Reasoning (COR) of university students and graduates. *Frontiers in Education*, 7:914857. doi: 10.3389/feduc.2022.914857

Nori, H., Lee, Y. T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., et al. (2023). Can generalist foundation models outcompete special-purpose tuning? Case study in medicine. *arXiv*, 1–21. doi: 10.48550/arXiv.2311.16452

OECD (2017). PISA 2015 results (volume V): collaborative problem solving. Paris: OECD Publishing.

OECD (2023). PISA 2025 learning in the digital world framework (second draft). OECD. Available at: https://www.oecd.org/media/oecdorg/satellitesites/pisa/PISA%202025%20Learning%20in%20the%20Digital%20World%20Assessment%20Framework%20-%20Second%20Draft.pdf (Accessed January 4, 2024).

Opara, E., Mfon-Ette Theresa, A., and Aduke, T. C. (2023). ChatGPT for teaching, learning and research: prospects and challenges. *Global Acad. J. Human. Soc. Sci.* 5, 33–40. doi: 10.36348/gajhss.2023.v05i02.001

Oppenlaender, J. (2022). A taxonomy of prompt modifiers for text-to-image generation. *Behav. Inform. Technol.*, 1–14. doi: 10.1080/0144929X.2023.2286532

Pellegrino, J. W. (2023). "Introduction: arguments in support of innovating assessments" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 15–28.

Pellegrino, J. W., Chudowsky, N., and Glaser, R. (2001). Knowing what students know: The science and Design of Educational Assessment. Washington, DC: National Academy Press.

Piacentini, M., Foster, N., and Nunes, C. A. A. (2023). "Next-generation assessments of 21st century competencies: insights from the learning sciences" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 45–60.

Prokop, M., Pilař, L., and Tichá, I. (2020). Impact of think-aloud on eye-tracking: a comparison of concurrent and retrospective think-aloud for research on decision-making in the game environment. *Sensors* 20:2750. doi: 10.3390/s20102750

Reid, M. Gemini Team Google (2024). Gemini 1.5: unlocking multimodal understanding across millions of tokens of context. *arXiv*, 1–154. doi: 10.48550/arXiv.2403.05530

Ridnik, T., Kredo, D., and Friedman, I. (2024). Code generation with AlphaCodium: from prompt engineering to flow engineering. *arXiv*, 1–10. doi: 10.48550/arXiv.2401.08500

Roll, I., and Barhak-Rabinowitz, M. (2023). "Measuring self-regulated learning using feedback and resources" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 159–171.

Sabatini, J., Hu, X., Piacentini, M., and Foster, N. (2023). "Designing innovative tasks and test environments" in Innovating assessments to measure and support complex skills. eds. N. Foster and M. Piacentini (Paris: OECD Publishing), 131–146.

Sahoo, P., Singh, A. K., Saha, S., Jain, V., Mondal, S., and Chadha, A. (2024). A systematic survey of prompt engineering in large language models: techniques and applications. *arXiv*, 1–9. doi: 10.48550/arXiv.2402.07927

Schmidt, S., Zlatkin-Troitschanskaia, O., Roeper, J., Klose, C., Weber, M., Bültmann, A.-K., et al. (2020). Undergraduate Students' Critical Online Reasoning—Process Mining Analysis. Frontiers in Psychology, 11:576273. doi: 10.3389/fpsyg.2020.576273

Semnani, S., Yao, V., Zhang, H., and Lam, M. (2023). WikiChat: stopping the hallucination of large language model chatbots by few-shot grounding on Wikipedia. *arXiv*, 1–27. doi: 10.48550/arXiv.2305.14292

Shah, C. (2024). From prompt engineering to prompt science with human in the loop. *arXiv*, 1–7. doi: 10.48550/arXiv.2401.04122

Shavelson, R. J., Zlatkin-Troitschanskaia, O., Beck, K., Schmidt, S., and Marino, J. (2019). Assessment of university students' critical thinking: next generation performance assessment. *Intern. J. Testing*. doi: 10.1080/15305058.2018.1543309

Sparks, J. R., Katz, I. R., and Beile, P. M. (2016). Assessing digital information literacy in higher education: a review of existing frameworks and assessments with recommendations for next-generation assessment. *ETS Res. Rep Series* 2016, 1–33. doi: 10.1002/ets2.12118

Surowiecki, J. (2005). The wisdom of crowds. *Anchor*. doi: 10.5555/1095645

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *arXiv*. doi: 10.48550/arXiv.1706.03762

Walter, Y. (2024). Embracing the future of artificial intelligence in the classroom: the relevance of AI literacy, prompt engineering, and critical thinking in modern education. *Int. J. Educ. Technol. High. Educ.* 21:15. doi: 10.1186/s41239-024-00448-3

Wang, Z. J., Chakravarthy, A., Munechika, D., and Chau, D. H. (2024). Wordflow: social prompt engineering for large language models. *arXiv*, 1–8. doi: 10.48550/arXiv.2401.14447

Wang, B., Rau, P. L. P., and Yuan, T. (2023a). Measuring user competence in using artificial intelligence: validity and reliability of artificial intelligence literacy scale. *Behav. Inform. Technol.* 42, 1324–1337. doi: 10.1080/0144929X.2022.2072768

Wang, J., Shi, E., Yu, S., Wu, Z., Ma, C., Dai, H., et al. (2023b). Prompt engineering for healthcare: methodologies and applications. *arXiv*, 1–33. doi: 10.48550/arXiv.2304.14670

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., et al. (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv*, 1–24. doi: 10.48550/arXiv.2203.11171

Weber, P., Pinski, M., and Baum, L. (2023). Toward an objective measurement of AI literacy. PACIS 2023 Proceedings, 60. Available at: https://aisel.aisnet.org/pacis2023/60 (Accessed December 22, 2023).

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., and Chi, E. (2022). Chain-ofthought prompting elicits reasoning in large language models. *arXiv*, 1–43. doi: 10.48550/arXiv.2201.11903V

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., et al. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv*, 1–19. doi: 10.48550/arXiv.2302.11382

Wineburg, S., Breakstone, J., McGrew, S., Smith, M. D., and Ortega, T. (2022). Lateral reading on the open internet: a district-wide field study in high school government classes. *J. Educ. Psychol.* 114, 893–909. doi: 10.1037/edu0000740

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., et al. (2023). Large language models as optimizers. *arXiv*, 1–42. doi: 10.48550/arXiv.2309.03409

Yao, Y., Li, Z., and Zhao, H. (2023). Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv*, 1–21. doi: 10.48550/arXiv.2305.16582

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., et al. (2023). Tree of thoughts: deliberate problem solving with large language models. *arXiv*, 1–14. doi: 10.48550/arXiv.2305.10601

Yu, Z., Liu, X., Liang, S., Cameron, Z., Xiao, C., and Zhang, N. (2024). Don't listen to me: understanding and exploring jailbreak prompts of large language models. *arXiv*, 1–18. doi: 10.48550/arXiv.2403.17336

Zlatkin-Troitschanskaia, O., Hartig, J., Goldhammer, F., and Krstev, J. (2021). Students' online information use and learning progress in higher education–A critical literature review. 46, 1996–2021.

Zhang, P., Shao, N., Liu, Z., Xiao, S., Qian, H., Ye, Q., et al. (2024). Extending Llama-3's context ten-fold overnight. *arXiv*, 1–5. doi: 10.48550/arXiv.2404.19553

Zhou, A., Li, B., and Wang, H. (2024). Robust prompt optimization for defending language models against jailbreaking attacks. *arXiv*, 1–28. doi: 10.48550/arXiv.2401.17263

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., et al. (2022). Large language models are human-level prompt engineers. *arXiv*, 1–40. doi: 10.48550/arXiv.2211.01910

Zhu, Y., Zhu, M., Liu, N., Ou, Z., Mou, X., and Tang, J. (2024). LLaVA-phi: efficient multi-modal assistant with small language model. *arXiv*, 1–6. doi: 10.48550/arXiv.2401.02330