Unleashing the potential of prompt engineering for large language models

Banghao Chen[1], Zhaofeng Zhang[1], Nicolas Langrené[1*], Shengxin Zhu[2,1*]

[1]Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College, Zhuhai 519087, Guangdong, China.
[2]Research Center for Mathematics, Beijing Normal University, No.18, Jingfeng Road, Zhuhai 519087, Guangdong, China.

*Corresponding author(s). E-mail(s): nicolaslangrene@uic.edu.cn; shengxin.zhu@bnu.edu.cn;
Contributing authors: chenbanghao@u.nus.edu; zhangzf@umich.edu;

Abstract

This comprehensive review delves into the pivotal role of prompt engineering in unleashing the capabilities of Large Language Models (LLMs). The development of Artificial Intelligence (AI), from its inception in the 1950s to the emergence of advanced neural networks and deep learning architectures, has made a breakthrough in LLMs, with models such as GPT-4o and Claude-3, and in Vision-Language Models (VLMs), with models such as CLIP and ALIGN. Prompt engineering is the process of structuring inputs, which has emerged as a crucial technique to maximize the utility and accuracy of these models. This paper explores both foundational and advanced methodologies of prompt engineering, including techniques such as self-consistency, chain-of-thought, and generated knowledge, which significantly enhance model performance. Additionally, it examines the prompt method of VLMs through innovative approaches such as Context Optimization (CoOp), Conditional Context Optimization (CoCoOp), and Multimodal Prompt Learning (MaPLe). Critical to this discussion is the aspect of AI security, particularly adversarial attacks that exploit vulnerabilities in prompt engineering. Strategies to mitigate these risks and enhance model robustness are thoroughly reviewed. The evaluation of prompt methods is also addressed, through both subjective and objective metrics, ensuring a robust analysis of their efficacy. This review also reflects the essential role of prompt engineering in advancing AI capabilities, providing a structured framework for future research and application.

Keywords: prompt engineering, Large Language Models, AI-generated content, adversarial attacks, evaluation, AI agent, GPT-4, Vision-Language Models

1 Introduction

In recent years, a significant milestone in artificial intelligence research has been the progression of natural language processing capabilities, primarily attributed to Large Language Models (LLMs). Many popular models, rooted in the transformer architecture [1], undergo training on extensive datasets derived from web-based text. Central to their design is a self-supervised learning objective, which focuses on predicting subsequent words in incomplete sentences. Those models are called Artificial Intelligence-Generated Content (AIGC), and their ability to generate coherent and contextually relevant responses is a result of this training process, where they learn to associate words and phrases with their typical contexts.

LLMs operate by encoding the input text into a high-dimensional vector space, where semantic relationships between words and phrases are preserved. The model then decodes this representation to generate a response, guided by the learned statistical patterns [2]. The quality of the response can be influenced by various factors, including the prompt provided to the model, the model's hyperparameters, and the diversity of the training data.

These models, including LLMs such as the GPT series [3, 4] by OpenAI, along with many others (e.g. Gemini [5, 6] and Gemini (BARD) [7] by Google, Claude series by Anthropic [8, 9], and Llama series open-source model from Meta [10, 11]), have revolutionized tasks ranging from information extraction to the creation of engaging content [12]. In parallel, the development of multimodal large models (MMLMs) has introduced the ability to process and generate not just text, but also images, audio, and other forms of data, showcasing their flexibility and effectiveness. These models integrate multiple data modalities into a single framework, demonstrating strong capabilities in tasks such as image description and visual question answering (VQA). Early MMLMs include the DALL-E series [13–15], which can generate images from textual descriptions, and CLIP, which can understand and relate text and image data in a unified manner [16, 17]. More powerful models such as GPT-4o by OpenAI [18] and Claude 3.5 Sonnet by Anthropic [8, 9] excel in multimodal tasks involving text generation and understanding, integrating natural language processing with various forms of data to perform diverse and complex tasks. While numerous advanced models are currently capable of processing audio, the majority of accessible Application Programming Interfaces (APIs) remain focused on text and vision modalities. With the gradual introduction of audio APIs, a broad expansion of research in this modality can be expected [19]. The evolution of LLMs reflects significant strides in AI research, characterized by increasing model complexity, enhanced training methodologies, and broader application potentials. These advancements underline the critical role of prompt engineering in maximizing the utility and accuracy of these models, ensuring that they can effectively cater to diverse and dynamic user needs. While this survey is mainly focused on prompt engineering for LLMs, the inclusion of vision-language models (VLMs) offers a broader perspective, revealing the potential and challenges of prompt engineering in handling multimodal data. By integrating research from both types of models, we can gain a deeper understanding of the applications of prompt engineering and provide valuable insights for future research and practice.

In real applications, the prompt is the input of the model, and prompt engineering can result in significant output differences [20]. Modifying both the structure (e.g., altering length, arrangement of instances) and the content (e.g., phrasing, choice of illustrations, directives) of the prompt can exert a notable influence on the model's behavior [21, 22].

Prompt engineering refers to the systematic design and optimization of input prompts to guide the responses of LLMs, ensuring accuracy, relevance, and coherence in the generated output. This process is crucial in harnessing the full potential of these models, making them more accessible and applicable across diverse domains. Over time, prompt engineering has evolved from an empirical practice into a well-structured research domain. As illustrated in Figure 1, the historical progression of prompt engineering showcases significant milestones from the early days of structured

inputs in the 1950s to advanced methodologies such as chain-of-thought prompting [23] and self-consistency prompting [24] developed in recent years. This review will primarily focus on techniques emerging from the period of rapid development after 2017.

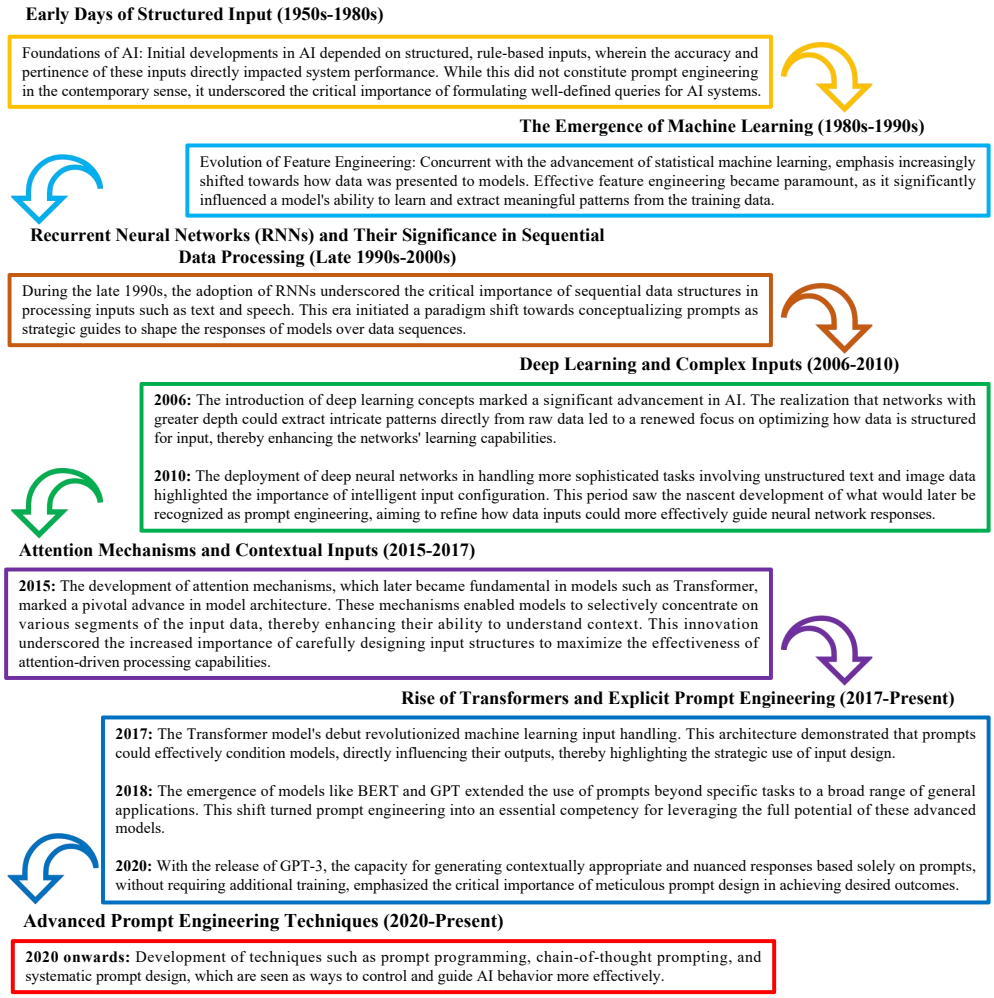

Fig. 1 History of the development in prompt engineering.

Contemporary prompt engineering encompasses a spectrum of techniques, ranging from foundational approaches such as role-prompting [25] to more sophisticated methods such as chain-of-thought prompting [23]. The domain remains dynamic, with emergent research continually unveiling novel techniques and applications in prompt engineering. The importance of prompt engineering is accentuated by its ability to guide model responses, thereby amplifying the versatility and relevance of LLMs in various sectors. Importantly, a well-constructed prompt can counteract challenges such as machine hallucinations [26, 27]. The influence of prompt engineering extends to numerous disciplines. For instance, it has facilitated the creation of robust feature extractors using LLMs, thereby improving their efficacy in tasks such as defect detection and classification [28].

This paper aims to provide a comprehensive review of the prompt engineering techniques proposed so far within the realm of LLMs. The structure of the paper is organized as follows: Section 2 explores the foundational methods of prompt engineering, emphasizing the importance of clear and precise instructions, role-prompting, and iterative attempts to optimize outputs. In Section 3, advanced methodologies such as chain-of-thought, self-consistency, and generated knowledge are introduced to

guide models in generating high-quality content. Section 4 discusses methodologies specific to VLMs, including Context Optimization (CoOp), Conditional Context Optimization (CoCoOp), and Multimodal Prompt Learning (MaPLe), which enhance the performance of VLMs [29]. Section 5 assesses the efficacy of various prompt methods through both subjective and objective evaluations, ensuring a robust analysis of their effectiveness. Section 6 briefly explores the applications of prompt engineering across diverse fields such as education, content creation, computer programming, and reasoning tasks, highlighting its broad impact. Section 7 addresses the security implications of prompt engineering, identifying common vulnerabilities in LLMs and reviewing strategies to enhance security such as adversarial training. Finally, Section 8 explores prospective methodologies, emphasizing the importance of understanding AI model structures and the potential of AI agents in advancing AI-generated content tools. This structured framework provides an entire overview of the pivotal role of prompt engineering in advancing AI capabilities and guiding future research and applications.

2 Basics of prompt engineering

By incorporating just a few key elements, one can craft a basic prompt that enables LLMs to produce high-quality answers. In this section, some essential components of a well-made prompt will be discussed and examples of these methods will be shown.

2.1 Model introduction: GPT-4

All of the examples in the following sections are generated by GPT-4, developed by OpenAI [4]. Vast amounts of text data have been used to train GPT-4, whose number of parameters has been estimated to be several orders of magnitude larger than the 175 billion parameters that had been used for the earlier GPT-3 [3]. The architectural foundation of the model rests on transformer [1], which essentially are attention mechanisms that assign varying weights to input data based on the context. Similar to GPT-3, GPT-4 was also fine-tuned to follow a broad class of written instructions by reinforcement learning from human feedback (RLHF) [30, 31], which is a technique that uses human preferences as a reward signal to fine-tune models.

When GPT-4 receives an input prompt, the input text will be firstly converted into tokens that the model can interpret and process. These tokens are then managed by transformer layers [32], which capture their relationships and context. Within these layers, attention mechanisms distribute different weights to tokens based on their relevance and context. After attention processing, the model forms its internal renditions of the input data, known as intermediate representations. These representations are then decoded back into human-readable text [33].

A significant aspect of this process is the randomness function [32]. This function is influenced by two primary parameters, temperature and top-k sampling. The first one, temperature [34] balances the randomness and determinism in the output. A higher temperature value results in more random outputs, while a lower value makes the output more deterministic. The second one, top-k sampling [35], limits the model's choices to the top k most probable tokens during each step of output generation. The final stage of this process is the output generation, where the model crafts the final text.

2.2 Giving instructions

The method of giving instructions, also known as re-reading [36], refers to the heuristics of human reading strategy. It has been observed that the output generated by GPT-4, introduced in Section 2.1, tends to be excessively general when provided with fundamental instructions devoid of any supplementary description [37, 38]. An example prompt is shown in Figure 2. When the model is prompted with a basic instruction, it faces a plethora of options, which makes the result quite broad. Hence, a comprehensive description is imperative to elicit more precise and relevant outputs [39].

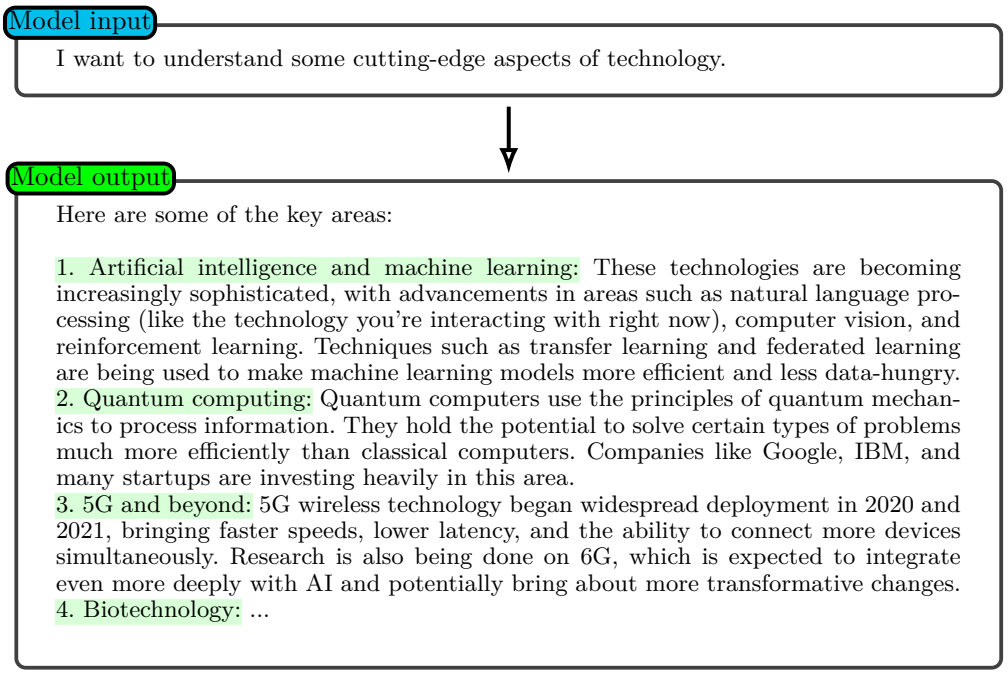

Fig. 2 Giving instructions without extra description.

2.3 Be clear and precise

The second basic prompt method is "to be clear and precise". This involves formulating prompts that are unambiguous and specific, which can guide the model toward generating the desired output.

Most LLM architectures are derived from an extensive array of textual data. It can be conceptualized as a combination of insights from a myriad of authors. When presented with a broad or undetailed prompt, its output predominantly exhibits a generic nature, which, while being applicable across a range of contexts, may not be optimal for any specific application. In contrast, a detailed and precise prompt enables the model to generate content that is more aligned with the unique requirements of the given scenario, as it reduces the model's uncertainty and guides it toward the correct response.

For instance, as shown in Figure 3, instead of asking a vague requirement such as "I want to understand the cutting edge of technology.", a more precise prompt would be "I want to understand the cutting edge of technology, specifically related to artificial intelligence and machine learning...".

2.4 Role-prompting

Role-prompting is another fundamental method in prompt engineering. It involves giving the model a specific role to play, such as a helpful assistant or a knowledgeable expert [40]. This method can be particularly effective in guiding the model's responses and ensuring that they align with the desired output. For instance, if the model is prompted to act as a historian, it is more likely to provide a detailed and contextually accurate response when asked about a historical event [41]. Another brief example is shown in Figure 4.

2.5 Use of triple quotes to separate

In prompt engineering, the use of triple quotes is a technique used to separate different parts of a prompt or to encapsulate multi-line strings. This technique is particularly useful when dealing with complex prompts that include multiple components or when the prompt itself contains quotes, which makes the model understand one's instructions better [42].

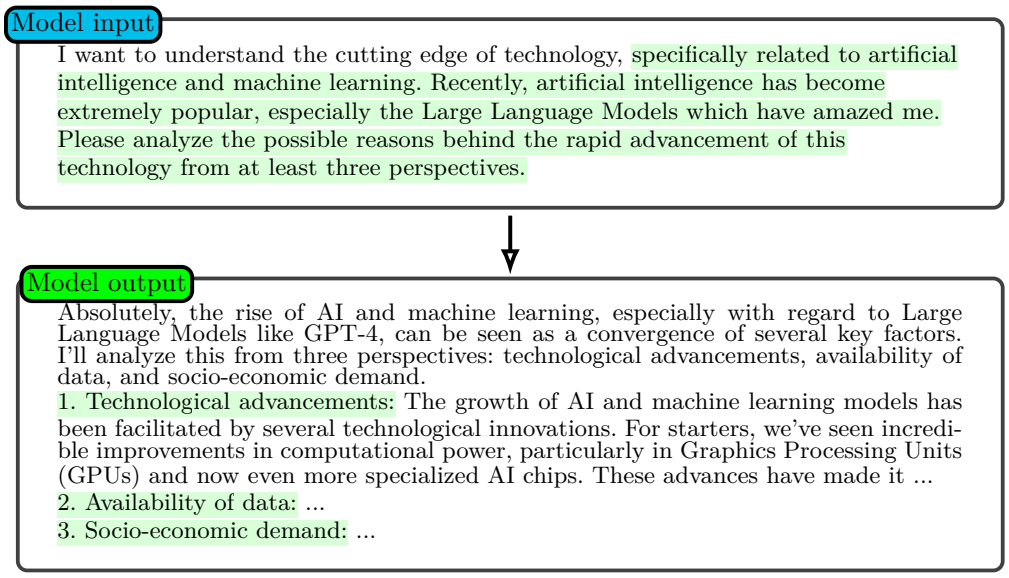

Fig. 3 A clearer and more precise prompt.

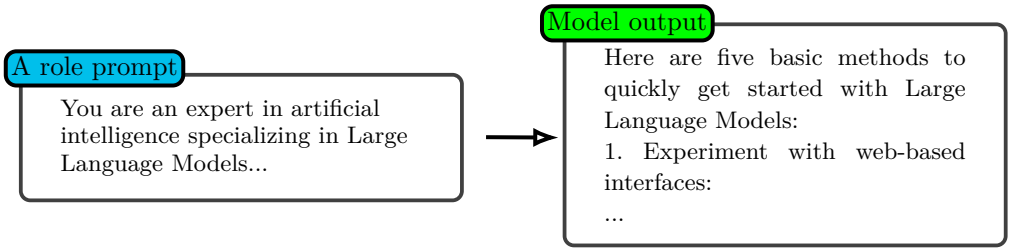

Fig. 4 Role prompting example.

2.6 Try several times

Due to the non-deterministic nature of LLMs, it is often beneficial to try several times when generating responses. This technique, often referred to as "resampling", involves running the model multiple times with the same prompt and selecting the best output. This approach can help overcome the inherent variability in the model's responses and increase the chances of obtaining a high-quality output [34].

2.7 One-shot or few-shot prompting

One-shot and few-shot prompting are two important techniques in prompt engineering. One-shot prompting refers to the method where the model is given a single example to learn from, while few-shot [43] prompting provides the model with multiple examples [44]. The choice between one-shot and few-shot prompting often depends on the complexity of the task and the capability of the model. For instance, for simple tasks or highly capable models, one-shot prompting might be sufficient. An example is shown in Figure 5. However, for more complex tasks or less capable models, few-shot prompting can provide additional context and guidance, thereby improving the model's performance.

However, as stated in [45], "examples don't always help", meaning that zero-shot prompting may have better output in some scenarios. Zero-shot prompting [46, 47], in the context of prompt-based learning, involves using a pre-trained LLM to perform tasks without any specific training for those tasks. The model relies on its general knowledge, acquired during pre-training, to generate predictions based on cleverly crafted prompts. This allows the LLMs to handle new tasks with no additional task-specific data, making it adaptable to scenarios with minimal labeled data. [45]

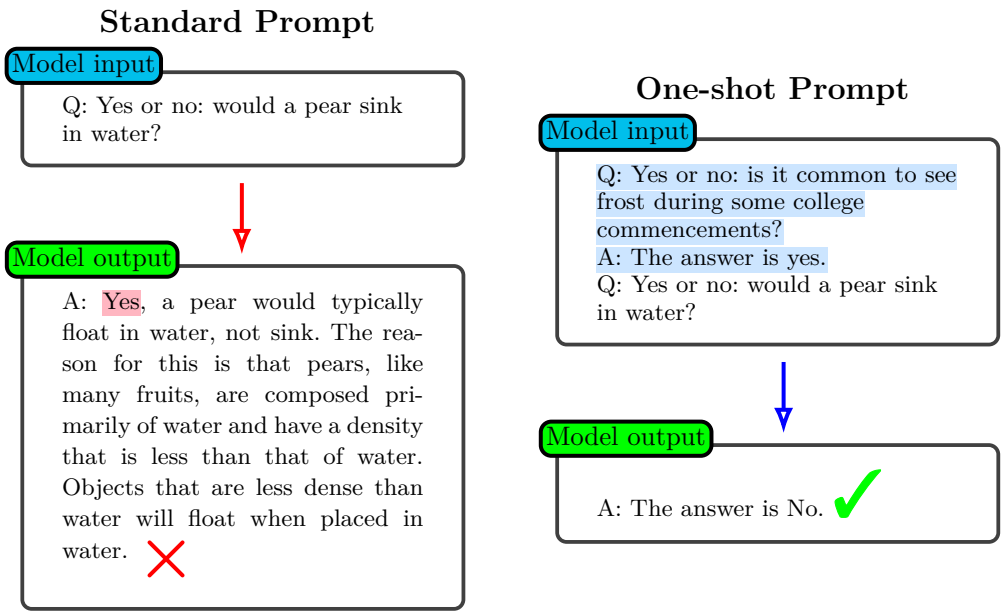

Fig. 5 Comparison of standard prompt and one-shot prompt.

investigated the intricacies of how large generative language models, such as GPT-3, respond to prompts. One of the significant findings from this paper is that zero-shot prompts can, in certain scenarios, outperform few-shot prompts. This suggests that the role of few-shot examples might not be as much about teaching the model a new task (meta-learning) but rather guiding it to recall a task it has already learned. This insight is crucial as it challenges the conventional wisdom that more examples always lead to better performance [3]. In the context of one-shot or few-shot prompting, it is essential to understand that while examples can guide the model, they do not always enhance its performance. Sometimes, a well-crafted zero-shot prompt can be more effective than providing multiple examples [48].

2.8 LLMs settings: temperature and top-p

The settings of LLMs, such as the temperature and top-p, play a crucial role in the generation of responses. The temperature parameter controls the randomness of the generated output: a lower temperature leads to more deterministic outputs [49, 50]. The top-p parameter, on the other hand, controls the nucleus sampling [34], which is a method to add randomness to the model's output [51]. Adjusting these parameters can significantly affect the quality and diversity of the model's responses, making them essential tools in prompt engineering. However, it has been noted that certain models, exemplified by ChatGPT, do not permit the configuration of these hyperparameters, barring instances where the Application Programming Interface (API) is employed. [52] ranks several AI text generators and text-to-image systems in terms of various openness metrics, including the accessibility of their API and model parameters.

3 Advanced methodologies

The foundational methods from the previous section can help us produce satisfactory outputs. However, experiments indicate that when using LLMs for complex tasks such as analysis or reasoning, the accuracy of the model's outputs still has room for improvement. In this section, advanced techniques of prompt engineering will be introduced to guide the model in generating more specific, accurate, and high-quality content.

3.1 Chain-of-thought

The concept of "Chain-of-Thought" (CoT) prompting [23] in LLMs is a relatively new development, which has been shown to significantly improve the accuracy of LLMs on various logical reasoning tasks [53–55]. CoT prompting involves providing intermediate reasoning steps to guide the model's responses, which can be facilitated through simple prompts such as "Let's think step by step" or through a series of manual demonstrations, each composed of a question and a reasoning chain that leads to an answer [56, 57]. It also provides a clear structure for the model's reasoning process, making it easier for users to understand how the model arrived at its conclusions.

[58] illustrates the application of CoT prompting to medical reasoning, showing that it can effectively elicit valid intermediate reasoning steps from LLMs. [59] introduces the concept of Self-Education via Chain-of-Thought Reasoning (SECToR), and argues that, in the spirit of reinforcement learning, LLMs can successfully teach themselves new skills by chain-of-thought reasoning. In another study, [60] used CoT prompting to train verifiers to solve math word problems, demonstrating the technique's potential in educational applications. [61] proposed a multimodal version of CoT, called Multimodal-CoT, to handle more complex, multimodal tasks beyond simple text-based tasks, such as visual tasks, further expanding the potential applications of CoT. Furthermore, many works are building upon the CoT framework, for instance, [62] introduces Automate-CoT, an automated approach for augmenting and selecting rationale chains to enhance the reasoning capabilities of LLMs, thereby reducing the reliance on manually crafted CoT prompts.

3.1.1 Zero-shot chain-of-thought

The concept of "Zero-Shot chain-of-thought" (Zero-shot-CoT) prompting is an advanced iteration of the CoT prompting mechanism, where the "zero-shot" aspect implies that the model is capable of performing some reasoning without having seen any examples of the task during training.

[47] discovered that the augmentation of queries with the phrase "Let's think step by step" facilitated the generation of a sequential reasoning chain by LLMs. This reasoning chain subsequently proved instrumental in deriving more precise answers. This technique is based on the idea that the model, much like a human, can benefit from having more detailed and logical steps to process the prompt and generate a response.

For instance, the standard prompt is illustrated in Figure 6, while the appended phrase, "Let's think step by step", is depicted in Figure 7. Observations indicate that the incorporation of "Let's think step by step" enhances the logical coherence and comprehensiveness of the model's response.

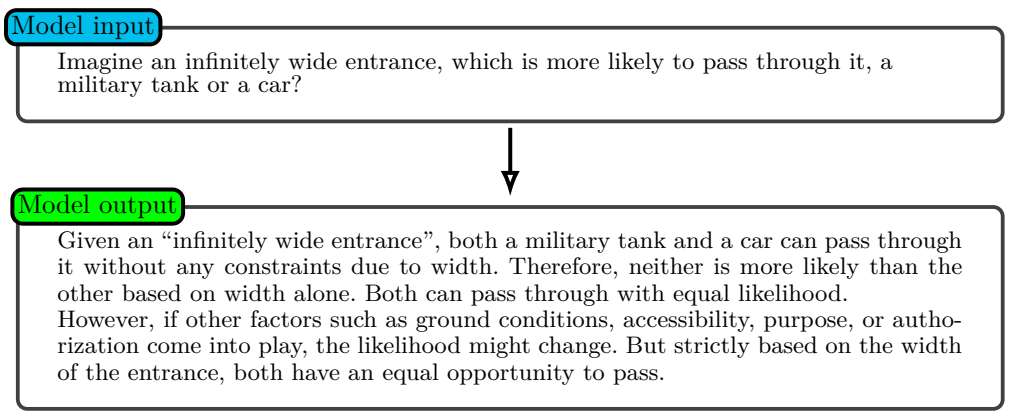

Fig. 6 Standard prompt.

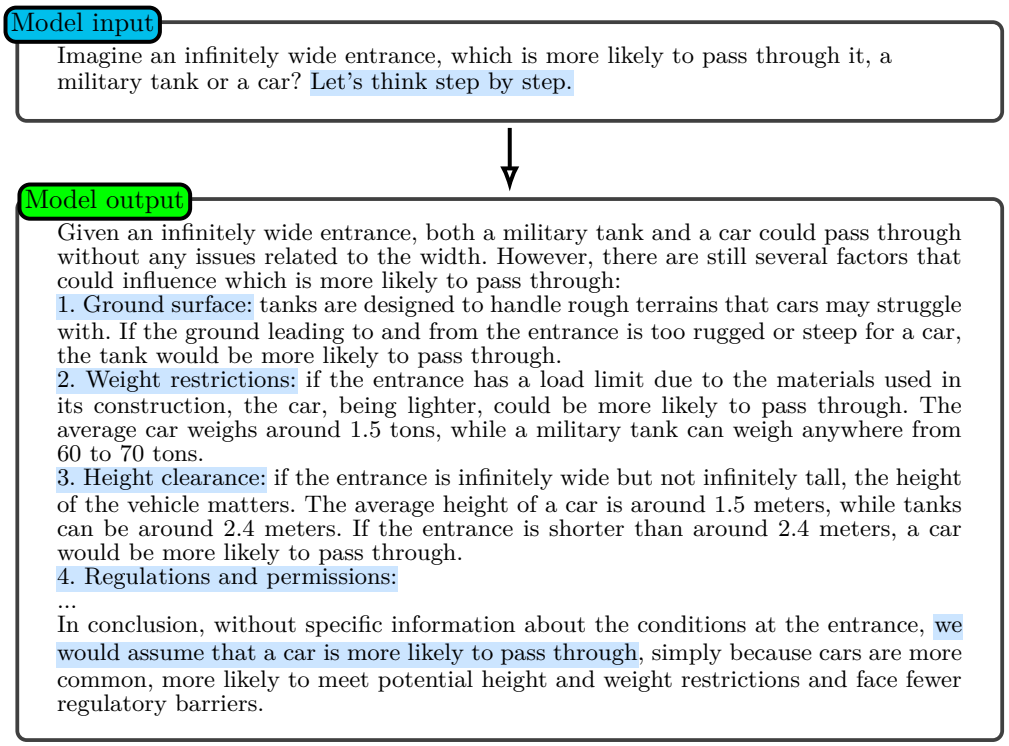

Fig. 7 Adding "Let's think step by step".

3.1.2 Golden chain-of-thought

[63] introduced the "golden chain-of-thought", providing an innovative approach to generating responses to instruction-based queries. This methodology leverages a set of "ground-truth chain-of-thought" solutions incorporated within the prompt, considerably simplifying the task for the model as it circumvents the necessity for independent CoT generation. Concurrently, a novel benchmark comprising detective puzzles has been designed, to assess the abductive reasoning capacities of LLMs, which is also considered an evaluation of the golden CoT. Finally, according to the experiment by [63], in the context of the golden CoT, GPT-4 exhibits commendable performance, boasting an 83% solve rate of puzzles in contrast to the 38% solve rate of the standard CoT.

However, the characteristics of the Golden CoT requiring the "ground-truth chain-of-thought solutions" as an integral part of the prompt also signifies that the Golden CoT's contribution to solving such problems is limited, despite its high solve rate of 83%.

3.2 Self-consistency

In the assessment of *Instruct*GPT [64] and GPT-3 [3] on a new synthetic QA dataset called Proof and Ontology-Generated Question-Answering (PrOntoQA) [65, 66], it was observed that although the most extensive model exhibited capability in reasoning tasks, it encountered challenges in proof planning and the selection of the appropriate proof step amidst multiple options, which caused accuracy uncertainties [65]. Self-consistency is one of the methods for LLMs to solve this situation, which is an advanced prompting technique that aims to ensure the model's responses are consistent with each other [23, 24]. This method greatly increases the odds of obtaining highly accurate results. The principle of self-consistency in language models posits that for a complex reasoning problem, there can be multiple reasoning paths leading to the correct answer. In this approach, a language model generates a diverse set of reasoning paths for the same problem. The most accurate and consistent answer is

then determined by evaluating and marginalizing across these varied paths, ensuring that the final answer reflects the convergence of multiple lines of thought.

The self-consistency method contains three steps. Firstly, prompt a language model using CoT prompting, then replace the "greedy decode" (1-Best) [32, 67] in CoT prompting by sampling from the language model's decoder to generate a diverse set of reasoning paths, and finally, marginalize out the reasoning paths and aggregate by choosing the most consistent answer in the final answer set.

It is noteworthy that self-consistency can be harmoniously integrated with most sampling algorithms, including but not limited to, temperature sampling [49, 50], top-k sampling [32, 68, 69], and nucleus sampling [34]. Nevertheless, such an operation may necessitate the invocation of the model's API to fine-tune these hyperparameters. In light of this, an alternative approach could be to allow the model to generate results employing diverse reasoning paths, and then generate a diverse set of candidate reasoning paths. The response demonstrating the highest degree of consistency across the various reasoning trajectories is then more inclined to represent the accurate solution [70].

[2, 71] have shown that self-consistency enhances outcomes in arithmetic, commonsense, and symbolic reasoning tasks. Furthermore, in practice, self-consistency can be combined with other techniques to further enhance the model's performance. [72] found that combining self-consistency with a discriminator-guided multi-step reasoning approach significantly improved the model's reasoning capabilities.

3.3 Generated knowledge

The "generated knowledge" [73] approach in prompt engineering is a technique that leverages the ability of LLMs to generate potentially useful information about a given question or prompt before generating a final response. This method is particularly effective in tasks that require commonsense reasoning, as it allows the model to generate and utilize additional context that may not be explicitly present in the initial prompt.

As exemplified in Figure 6, when posing the query to the model, "Imagine an infinitely wide entrance, which is more likely to pass through it, a military tank or a car?", standard prompts predominantly yield responses that neglect to factor in the "entrance height". Conversely, as delineated in Figure 8 and Figure 9, prompting the model to first generate pertinent information and subsequently utilizing generated information in the query leads to outputs with augmented logical coherence and comprehensiveness. Notably, this approach stimulates the model to account for salient factors such as "entrance height".

3.4 Least-to-most prompting

The concept of "least-to-most prompting" [74] is an advanced method that involves decomposing a complex problem into a series of simpler subproblems, which are then addressed sequentially. The foundational premise of this approach is to systematically break down intricate tasks into manageable components. Each subproblem is solved in turn, with the solution to each one serving as a building block for the next. This step-by-step process not only simplifies the problem-solving task for the language model but also enables it to handle more complex challenges than those typically managed by traditional prompting techniques. Figure 10 is an illustration of least-to-most prompting applied to a mathematical word. This figure illustrates the least-to-most prompting approach by systematically breaking down a complex problem into a series of simpler subproblems. The process begins with the decomposition of the main problem—calculating the distance a train travels in 2.5 hours—into two sequential subproblems. First, the model is prompted to determine the train's speed, and then it uses this information to calculate the distance traveled. Each subproblem is solved in sequence, with the solution to the first subproblem feeding into the second. The solutions are then aggregated to arrive at the final answer. This method emphasizes

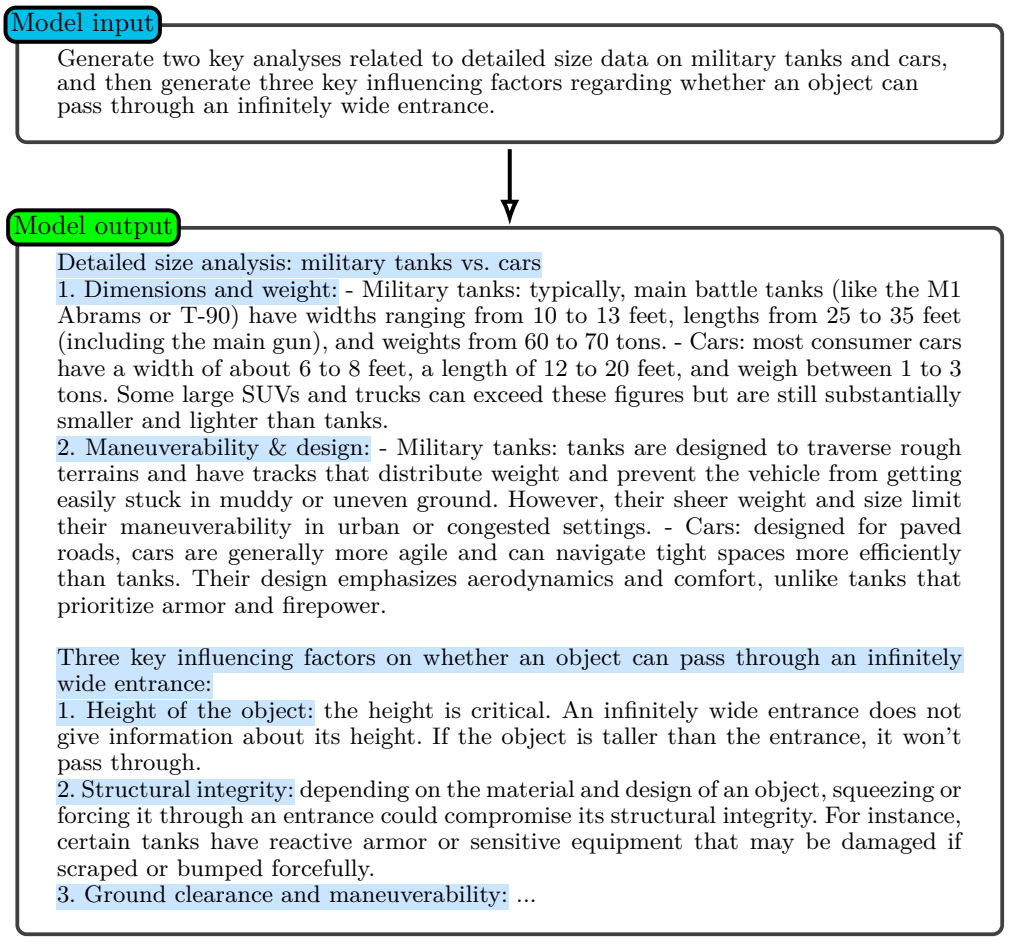

Fig. 8 Generating knowledge (Step1).

the key principles of problem decomposition and sequential problem solving, enabling the model to manage and solve complex tasks more effectively.

Upon rigorous experimentation in domains including symbolic manipulation, compositional generalization, and mathematical reasoning, [74] substantiate that the least-to-most prompting paradigm exhibits the capacity to generalize across challenges of greater complexity than those initially presented in the prompts. [75] introduced Program Aided Language models (PAL), using the LLMs to read natural language problems and generate programs as the intermediate reasoning steps. By using least to most prompting, PAL shows enhancement on GSM8K [60] and SVAMP [76], which are benchmarks about complex mathematical problems for LLMs.

3.5 Tree of thoughts

The "tree of thoughts" (ToT) prompting technique in LLMs is an advanced method that employs a structured approach to guide LLMs in their reasoning and response generation processes. It enhances problem-solving by exploring multiple reasoning paths, termed 'thoughts'. Unlike traditional linear prompts, ToT allows LLMs to consider various possible solutions and strategies, including looking ahead, backtracking, and self-evaluation, making it more interactive and adaptable to the complexity of the task at hand. This approach fosters more dynamic and deliberate decision-making in complex problem-solving tasks, moving beyond a rigid, hierarchical prompt structure to a more flexible and adaptive reasoning process [77]. The ToT approach, for instance, when applied to complex mathematical problem-solving, prompts the model to generate various potential solutions and evaluate them, rather than simply asking for a solution.

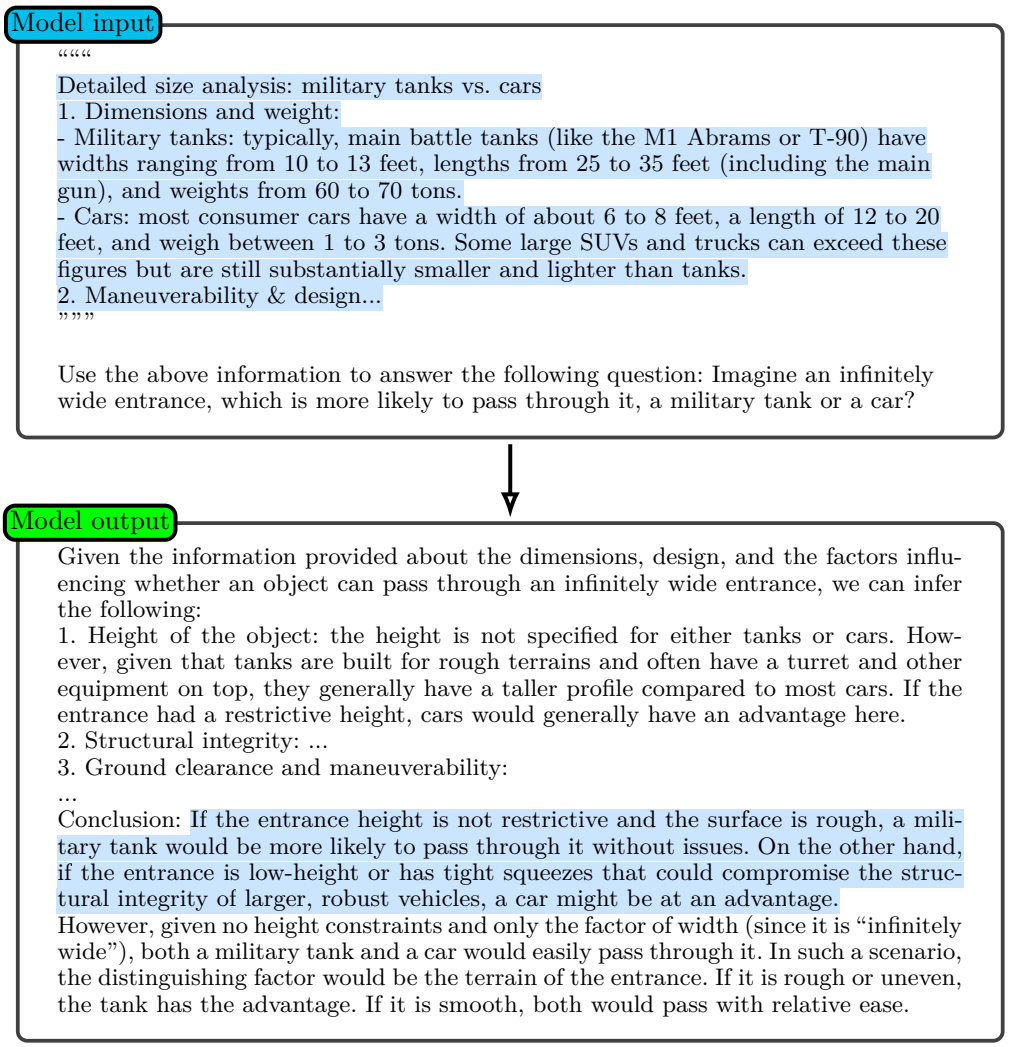

Fig. 9 Combining the generated knowledge with the question (Step2).

[77] demonstrates that this formulation is more versatile and can handle challenging tasks where standard prompts might fall short. Another research by [78] further emphasizes the potential of this technique in enhancing the performance of LLMs by structuring their thought processes.

[7] introduces the “tree-of-thought prompting”, an approach that assimilates the foundational principles of the ToT frameworks and transforms them into a streamlined prompting methodology. This technique enables LLMs to assess intermediate cognitive constructs within a singular prompt. An exemplar ToT prompt is delineated in Figure 11.

3.6 Graph of thoughts

Unlike the “chain-of-thoughts” or “tree of thoughts” paradigms, the “graph of thoughts” (GoT) framework [79] offers a more intricate method of representing the information generated by LLMs. The core concept behind GoT is to model this information as an arbitrary graph. In this graph, individual units of information, termed “LLM thoughts”, are represented as vertices. The edges of the graph, on the other hand, depict the dependencies between these vertices. This unique representation allows for the combination of arbitrary LLM thoughts, thereby creating a synergistic effect in the model’s outputs.

In the context of addressing intricate challenges, LLMs utilizing the GoT framework might initially produce several autonomous thoughts or solutions. These

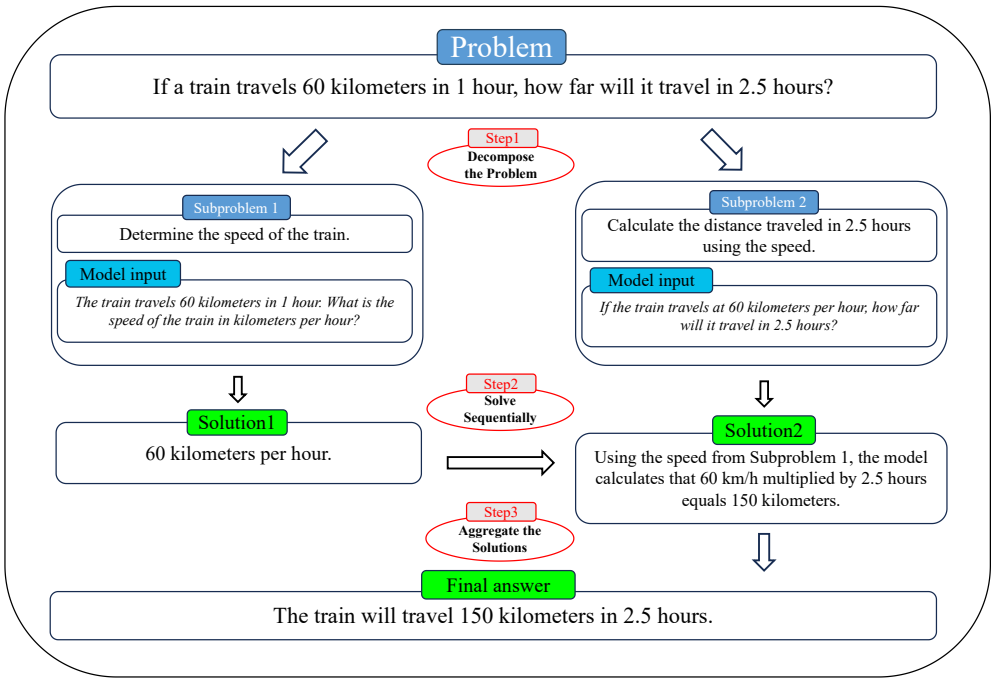

Fig. 10 Illustration of Least-to-Most Prompting Applied to a Mathematical Word.

Tree of thoughts prompting

Imagine three different experts answering this question.
All experts will write down 1 step of their thinking,
then share it with the group.
Then all experts will go on to the next step, etc.
If any expert realizes they're wrong at any point then they leave.
The question is...

Fig. 11 A sample ToT prompt [7].

individual insights can subsequently be interlinked based on their pertinence and interdependencies, culminating in a detailed graph. This constructed graph permits diverse traversal methods, ensuring the final solution is both precise and comprehensive, encompassing various dimensions of the challenge.

The efficacy of the GoT framework is anchored in its adaptability and the profound insights it can yield, particularly for intricate issues necessitating multifaceted resolutions. Nonetheless, it is imperative to recognize that while GoT facilitates a systematic approach to problem-solving, it also necessitates a profound comprehension of the subject matter and meticulous prompt design to realize optimal outcomes [80].

3.7 Decomposed prompting

Decomposed Prompting (DECOMP) [81] is a modular approach designed to tackle complex tasks by breaking them down into simpler, manageable sub-tasks. This methodology leverages the capabilities of LLMs by creating a systematic process where each sub-task is handled by specialized handlers. The approach not only simplifies the problem-solving process but also enhances the flexibility and efficiency of task handling.

Four key components of this method are shown in Figure 12. The core of DECOMP involves a decomposer LLM that generates a prompting program P for a complex task Q. The program P is a sequence of steps, each step directing a simpler sub-query to a function within an auxiliary set of sub-task functions F. The program can be

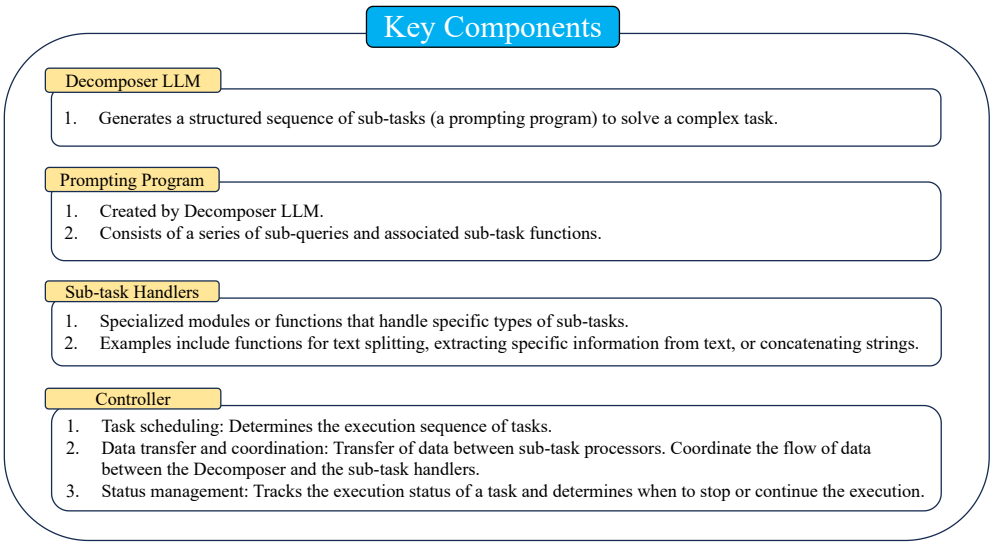

Fig. 12 Key components of DECOMP

represented as:

$$P = \{(f_1, Q_1, A_1), \ldots, (f_k, Q_k, A_k)\}$$

where A_k is the final answer predicted by P, and Q_i is a sub-query directed to the sub-task function $f_i \in F$. A high-level imperative controller manages the execution of P, passing inputs and outputs between the decomposer and sub-task handlers until the final output is obtained.

To teach the decomposer LLM, in-context examples are used. These examples demonstrate the decomposition of complex queries into simpler sub-queries. Each example E_j takes the form:

$$E_j = \left(Q_j, \left\{(f_{j,1}, Q_{j,1}, A_{j,1}), \ldots, (f_{j,k_j}, Q_{j,k_j}, A_{j,k_j})\right\}\right)$$

where $A_{j,k_j} = A_j$ is the final answer for Q_j, and $(Q_{j,1}, \ldots, Q_{j,k_j})$ represents the decomposition of Q_j. Each sub-task function f is operationalized through sub-task handlers, which can be additional LLM prompts or symbolic or learned functions [81]. An illustration of the process flow is shown in Figure 13.

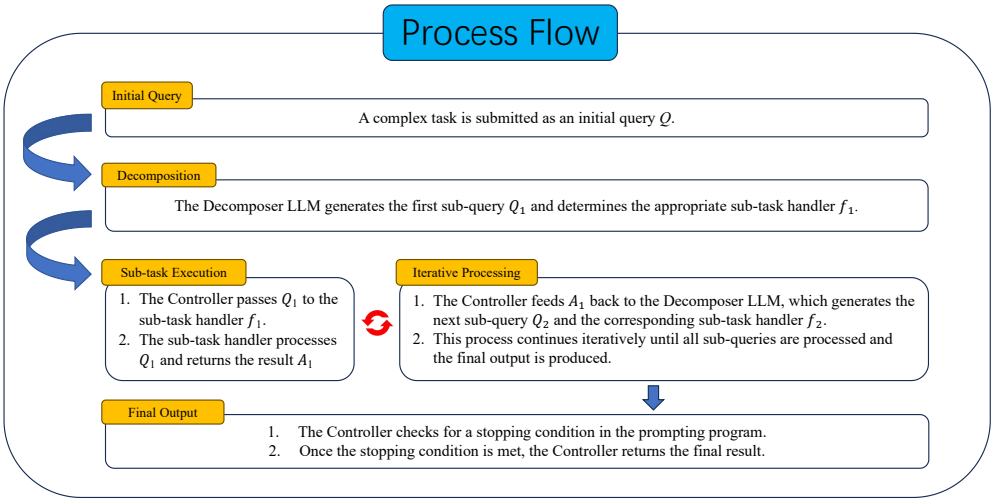

Fig. 13 An example of the process flow of DECOMP.

The DECOMP approach has several advantages. First, its modularity allows each sub-task handler to be independently optimized, debugged, and upgraded, which facilitates systematic performance improvements and easier integration of new methods or models. Second, DECOMP can incorporate error-correcting sub-task handlers, improving the overall accuracy and reliability of the system. Third, the approach allows for diverse decomposition structures, including hierarchical and recursive decompositions, which are particularly useful for handling complex and large-scale problems. Finally, sub-task handlers can be shared across different tasks, enhancing the efficiency of the problem-solving process.

DECOMP and Least-to-Most Prompting [74] both decompose complex tasks to enhance large language models' reasoning abilities, but DECOMP distinguishes itself through its flexible, modular approach. Unlike Least-to-Most Prompting's linear progression from easy to hard sub-questions, DECOMP allows for non-linear and recursive decomposition, with dedicated sub-task handlers that can be independently optimized and replaced. This modularity not only enhances flexibility and reusability across tasks but also introduces potential error-correcting mechanisms, making DECOMP more robust and adaptable to complex, multi-step reasoning tasks. While DECOMP has demonstrated superior performance in specific domains, such as symbolic reasoning and multi-step question answering, its advantages over Least-to-Most Prompting may vary depending on the nature of the task [81].

In case studies, DECOMP demonstrated superior performance in various scenarios. For instance, in the k-th letter concatenation task, DECOMP outperformed CoT prompting by effectively teaching the sub-task of extracting the k-th letter through further decomposition. In list reversal, DECOMP showed better length generalization compared to CoT by recursively decomposing the task into reversing smaller sub-lists, achieving higher accuracy for longer input sequences. In long-context question answering (QA), DECOMP allowed for handling more examples than feasible with CoT prompting, leading to improved performance. In open-domain QA, incorporating symbolic retrieval APIs within the DECOMP framework enhanced performance on multi-hop QA datasets compared to CoT prompting. Additionally, in Math QA, DECOMP improved accuracy by post-processing CoT prompts to fix frequent formatting errors, resulting in significant performance gains [81].

By leveraging the modular, flexible, and systematic approach of DECOMP, complex tasks can be effectively decomposed and solved, showcasing its superiority over traditional CoT prompting and other contemporary methods.

3.8 Active prompt

The adoption of the active prompt [82] method marks a significant advancement in the utilization of LLMs for complex reasoning tasks. The active prompt method does not involve the traditional process of prefix-tuning [83]. Instead, it focuses on improving the reasoning capabilities of LLMs through strategic selection and annotation of task-specific examples. By systematically selecting and annotating the most uncertain questions, this method not only refines the model's understanding but also leverages human expertise more effectively [84]. The process begins with the generation of multiple predictions for each question, followed by the calculation of uncertainty (uncertainty estimation) [85, 86] using various metrics such as disagreement, entropy, and variance. This strategic selection process ensures that the most informative questions are prioritized for annotation. The human annotation phase is crucial, as it involves providing detailed chain-of-thought reasoning and answers, which are then used to prompt the LLM during inference. This annotated data serves as exemplars, guiding the model through complex reasoning pathways and enhancing its predictive accuracy. The application of self-consistency [24] techniques further solidifies the model's reliability by selecting the most consistent answers from multiple reasoning paths. The key innovation of this method is the thought of finding out the most efficient one-shot or few-shot [43] examples, so it improves the inference ability of specific fields. A concrete process illustration is shown in Figure 14.

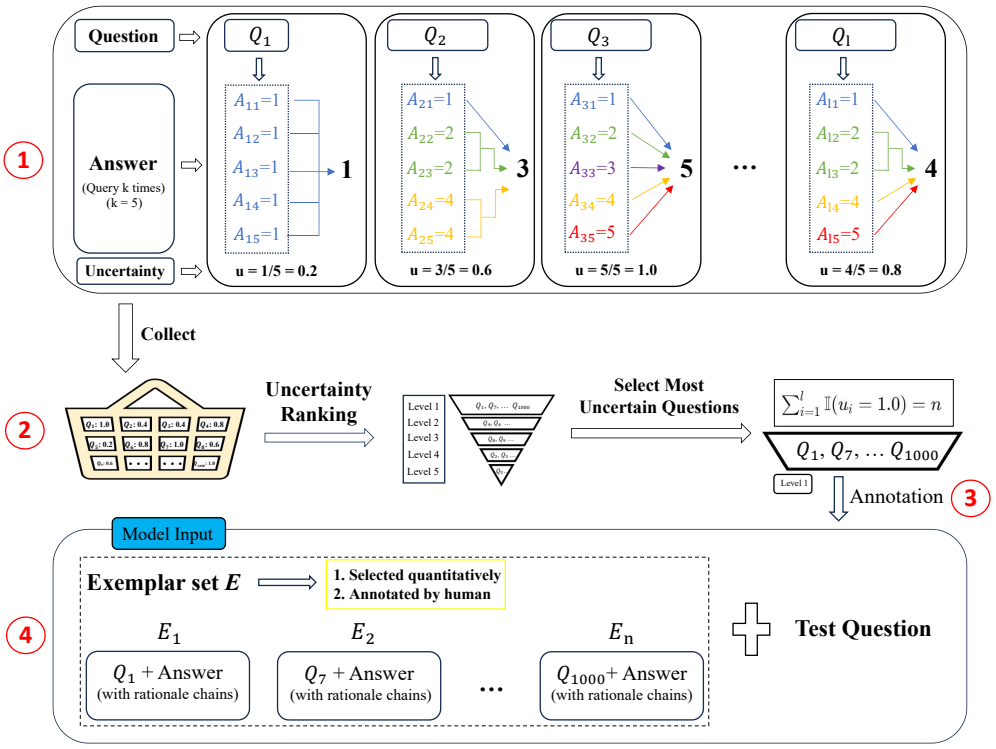

Fig. 14 Illustration of the whole process. (1) Uncertainty Estimation. (2) Collection, Ranking and Selection. (3) Annotation (by human). (4) Inference.

The active prompt method offers several key benefits, including efficient task adaptation and significant performance improvements across various reasoning domains. This approach aligns with the broader trend towards more interactive and adaptive AI systems, emphasizing the importance of responsive design in prompt engineering. Its ability to reduce human engineering efforts by focusing on the most uncertain and informative questions makes it an important tool for advancing LLM capabilities. This method not only enhances the quality of task-specific prompts but also maximizes the use of human expertise, paving the way for more sophisticated and accurate AI systems [82].

3.9 Prompt pattern catalog

A Prompt Pattern Catalog [87] is an organized collection of prompt templates and patterns designed to enhance the effectiveness of prompt engineering, particularly for LLMs such as ChatGPT. This methodology involves creating a standardized set of prompt patterns that can be applied across various tasks, ensuring consistency and optimizing the performance of models through systematic prompt design. By developing a catalog of prompt patterns, researchers and practitioners can ensure a consistent approach to prompt engineering, reducing variability and errors from ad hoc prompt creation [87, 88]. Predefined prompt patterns streamline the process of prompt engineering, saving time and resources by allowing practitioners to select and adapt patterns rather than crafting new prompts from scratch. A well-designed prompt pattern catalog includes patterns for various contexts and applications, enabling models to be quickly adapted to new tasks and domains by selecting the most appropriate patterns. Systematic use of optimized prompt patterns enhances model performance by providing more effective and contextually appropriate prompts, leading to better task-specific results [89].

The central methodology of this research involves the conceptualization and application of prompt patterns, which are reusable solutions to common problems encountered when interacting with LLMs. These prompt patterns are analogous

to design patterns in software engineering, providing structured and documented approaches to enhance the output and interaction quality of LLMs. The framework for documenting these prompt patterns includes a detailed structure that ensures their adaptability across different domains [87].

To systematically categorize these prompt patterns, the authors have divided them into five primary categories: Input Semantics, Output Customization, Error Identification, Prompt Improvement, and Interaction. This classification helps in organizing the patterns based on their functional roles and the specific problems they address. Within this framework, the research introduces a comprehensive catalog of 16 distinct prompt patterns. Each pattern is meticulously documented with the following components: name and classification, intent and context, motivation, structure and key ideas, example implementation, and practical consequences. The prompt patterns cover a wide range of functionalities. For instance, the Input Semantics category includes patterns such as Meta Language Creation, which helps in defining custom input languages for LLMs. The Output Customization category features patterns such as Output Automater and Visualization Generator, which tailor the generated outputs to specific formats or visualizations. Error Identification patterns such as Fact Check List ensure the accuracy of generated content by highlighting critical facts for verification. Prompt Improvement patterns, including Question Refinement and Alternative Approaches, enhance the quality of interactions by refining questions and suggesting multiple ways to achieve a goal. Lastly, Interaction patterns such as Flipped Interaction and Game Play facilitate dynamic and engaging user-LLM interactions [87].

The methodology also emphasizes the combinatory use of these patterns to tackle more complex prompt engineering tasks. By providing detailed examples and practical implementations, the research demonstrates how multiple prompt patterns can be integrated to create sophisticated and efficient prompting strategies. This structured approach not only improves the effectiveness of LLMs in various applications but also contributes to the broader understanding and advancement of prompt engineering as a field [87].

Research supports the effectiveness of prompt pattern catalogs. [87] outlines that the development and use of a prompt pattern catalog can improve the effectiveness and efficiency of prompt engineering with LLMs. [89] explores how predefined structured prompt patterns can enhance user interaction and improve model outputs in conversational AI. [88] investigates the application of prompt engineering patterns in enterprise settings, demonstrating their utility in optimizing model performance across various tasks. Additionally, [90] highlights the benefits of using predefined structured prompt patterns in software development, demonstrating significant improvements in code quality, requirements elicitation, and refactoring efficiency.

3.10 Prompt optimization

In the domain of prompt engineering for LLMs, the challenge of crafting effective prompts remains a significant barrier due to the extensive manual effort and expertise required. Prompt optimization is a critical technique for improving the performance of LLMs by refining the input prompts that guide their responses. The process of prompt optimization systematically adjusts these prompts to enhance accuracy and relevance, reducing the need for manual trial and error.

Several methods have been developed to automate prompt optimization, including gradient-based approaches such as Prompt Optimization with Textual Gradients (ProTeGi) [91], which uses text-based gradients to iteratively refine prompts, and black-box methods that optimize prompts based solely on output performance without requiring model internals. Additionally, model-adaptive techniques, such as Model-Adaptive Prompt Optimization (MAPO) [92], tailor the optimization to the specific characteristics of the LLM, potentially offering superior results. Each method has its advantages: gradient-based techniques are efficient and directed, black-box approaches are broadly applicable and easy to implement, and model-adaptive methods provide

customized optimization for specific models. The choice of method depends on task requirements, model complexity, and available resources.

3.10.1 Prompt optimization with textual gradients

Prompt Optimization with Textual Gradients (ProTeGi) [91] is inspired by gradient descent, a fundamental technique in optimization, but adapts this concept to the discrete and non-parametric nature of natural language processing. Instead of relying on numerical gradients, ProTeGi generates "textual gradients", which are natural language descriptions of the flaws in a given prompt based on its performance on a small batch of data. These gradients indicate the semantic direction in which the prompt needs to be improved.

ProTeGi further enhances this optimization process by applying these textual gradients to modify the prompt in the opposite semantic direction, akin to a reverse gradient descent in the language space. This iterative process is guided by a beam search algorithm combined with a bandit selection strategy, which efficiently explores the space of possible prompts and selects the most promising candidates for further refinement [91].

The effectiveness of ProTeGi has been demonstrated across multiple NLP tasks, including sentiment analysis, fake news detection, and the novel problem of LLM jailbreak detection. Experimental results indicate that ProTeGi can significantly improve prompt performance, with reported gains of up to 31% over initial prompts, while also surpassing existing prompt optimization methods in efficiency and accuracy. This method provides a robust, data-driven approach to prompt engineering, offering a scalable solution that can adapt to various tasks without requiring access to the internal states of LLMs [91].

3.10.2 Black-box prompt optimization

In recent prompt engineering research, the challenge of aligning LLMs with human intent without model retraining has garnered significant attention. Traditional alignment techniques, such as Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO), typically require substantial computational resources and direct access to model parameters, which are not always feasible or efficient, particularly with closed-source models such as GPT-4 or Claude-2. In response to these limitations, a novel method called Black-box Prompt Optimization (BPO) [93] has been introduced, providing a promising alternative for enhancing model alignment through prompt optimization alone.

BPO shifts the focus from model-centric to input-centric optimization, where the key idea is to refine the user's prompts rather than altering the model's internal parameters. This approach leverages feedback from pre-existing datasets that contain human preferences, creating pairs of original and optimized prompts. These pairs are then used to train a sequence-to-sequence model designed to rewrite prompts in a way that improves the alignment of LLM outputs with human expectations [93].

The BPO method offers several advantages. First, it is model-agnostic, allowing it to be applied across various LLMs, whether open-source or API-based, without requiring access to the model's internals. Second, it enhances interpretability, as the changes made to prompts are transparent and directly observable, providing clear insights into how and why a particular prompt leads to better alignment. Third, empirical results demonstrate that BPO not only improves the alignment of models such as GPT-3.5 and LLaMA-2 but also outperforms RLHF and DPO when used independently or in conjunction with these methods [93].

3.10.3 Model-adaptive prompt optimization

Traditionally, prompt optimization has focused on tailoring prompts to specific tasks to enhance model performance. However, [92] highlighted the necessity of adapting prompts not just to tasks but also to the specific characteristics of different LLMs. This shift in perspective has led to the development of Model-Adaptive Prompt

Optimization (MAPO), a novel approach designed to fine-tune prompts for individual LLMs, thereby maximizing their effectiveness across various downstream tasks. MAPO addresses the inherent variability in how different LLMs respond to the same prompt by introducing a two-phase optimization process. The first phase involves establishing a warm-up dataset, where candidate prompts are generated and evaluated for their suitability to each LLMs. This is followed by a combination of Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL), particularly employing techniques such as Proximal Policy Optimization (PPO) and Ranking Responses from Model Feedback (RRMF). This joint learning approach refines the prompts, ensuring they align with the specific preferences of each LLMs.

Empirical studies demonstrate that MAPO significantly improves performance in tasks such as question-answering, classification, and text generation when compared to conventional task-specific prompt optimization methods. By shifting the focus from a one-size-fits-all approach to a more nuanced, model-specific strategy, MAPO enhances the robustness and generalization of LLMs, making it a powerful tool in the prompt engineering toolkit [92].

3.10.4 PromptAgent

The PromptAgent method suggests framing prompt optimization as a strategic planning problem. A kind core of this method is the use of Monte Carlo Tree Search (MCTS), a principled planning algorithm that strategically navigates the vast space of expert-level prompts. Unlike conventional methods that generate prompts through local variations, PromptAgent employs a trial-and-error mechanism, inspired by human problem-solving strategies. This approach allows the model to iteratively refine prompts based on error feedback, simulating future rewards and prioritizing high-reward paths [94]. Another core of this method, for instance, PromptSource [95], collects over 2,000 open-source prompts for roughly 170 datasets, by dataset exploration, prompt writing and documentation to provide an enhanced prompt.

PromptAgent's effectiveness has been demonstrated across a diverse set of tasks, spanning general NLP challenges and domain-specific applications such as biomedical text processing. By autonomously generating prompts that incorporate domain-specific knowledge and detailed task instructions, PromptAgent consistently outperforms both human-designed prompts and other automated optimization methods [94], highlighting the importance of integrating strategic planning and self-reflection capabilities into prompt optimization frameworks.

3.10.5 Reinforcement learning

Reinforcement Learning (RL) for prompt optimization is an advanced technique designed to enhance the performance of LLMs by iteratively refining the prompts used during training and inference. This method utilizes the principles of reinforcement learning to navigate the complex parameter space of large models, optimizing the prompts for improved task-specific performance. In RL for prompt optimization, a reward function is defined to evaluate the effectiveness of different prompts based on the model's output. The model then uses this feedback to adjust and optimize the prompts through a series of iterations, ensuring that the prompts evolve to maximize performance on the target task by leveraging the model's ability to learn from its interactions with the environment [96].

Consider the task of VQA, where the goal is to generate accurate answers to questions based on visual input. Using RL for prompt optimization, the model can start with a set of initial prompts and iteratively refine them based on the accuracy of the generated answers. For instance, if the model is asked, "What is the color of the car in the image?" the initial prompts might produce varied responses. The reward function will assess these responses, favoring prompts that lead to correct answers. Over multiple iterations, the model learns to generate more precise prompts, improving its ability to accurately answer similar questions in the future [97].

3.10.6 GPTs (plugins)

Before ending this discussion on prompt optimization techniques, we need to mention the use of external prompt engineering assistants that have been developed recently and exhibit promising potential. Unlike the methods introduced previously, these instruments can help us to polish the prompt directly. They are adept at analyzing user inputs and subsequently producing pertinent outputs within a context that is defined by itself, thereby amplifying the efficacy of prompts. Some of the plugins provided by the OpenAI GPT store are good examples of such tools [98]. Some popular GPT store apps that specialize in generating or optimizing prompts are shown in Figure 15.

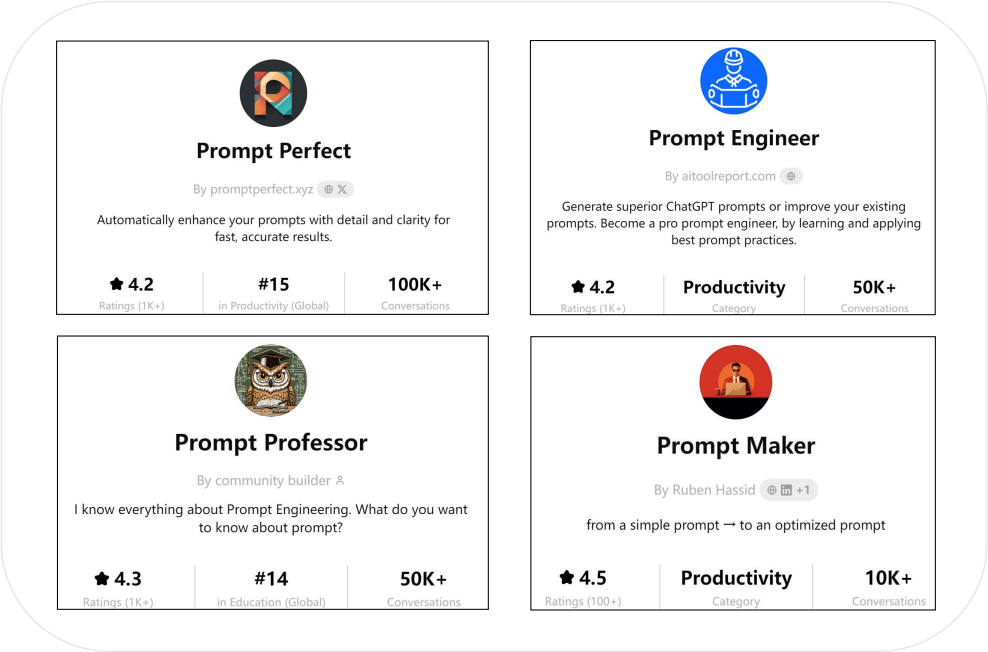

Fig. 15 Examples of GPT apps that specialize in generating or optimizing prompts [99].

In certain implementations, the definition of a plugin is incorporated into the prompt, altering the output [100]. Such integration may impact the manner in which LLMs interpret and react to the prompts, illustrating a connection between prompt engineering and plugins. Plugins mitigate the laborious nature of intricate prompt engineering, enabling the model to more proficiently comprehend or address user inquiries without necessitating excessively detailed prompts. Consequently, plugins can bolster the efficacy of prompt engineering while promoting enhanced user-centric efficiency. These tools, akin to packages, can be seamlessly integrated into Python and invoked directly [101, 102]. For instance, the "Prompt Enhancer" pluging [103], developed by AISEO [104], can be invoked by starting the prompt with the word "AISEO" to let the AISEO prompt generator automatically enhance the LLM prompt provided. Similarly, another plugin called "Prompt Perfect", can be used by starting the prompt with 'perfect' to automatically enhance the prompt, aiming for the "perfect" prompt for the task at hand [105, 106]. Nevertheless, while the use of plugins to enhance prompts is simple and handy, it is not always clear which prompt engineering technique, or combination of techniques, is implemented by a given plugin, given the closed-source nature of most plugins.

3.11 Retrieval augmentation

Another direction of prompt engineering research is to aim to reduce hallucinations. When using AIGC tools such as GPT-4, it is common to face a problem called "hallucinations", which refer to the presence of unreal or inaccurate information in the model's generated output [26, 107]. While these outputs may be grammatically correct, they can be inconsistent with facts or lack real-world data support. Hallucinations arise because the model may not have found sufficient evidence in its training data to support its responses, or it may overly generalize certain patterns when attempting to generate fluent and coherent output [108].

An approach to reduce hallucinations and enhance the effectiveness of prompts is the so-called retrieval augmentation technique, which aims at incorporating up-to-date external knowledge into the model's input [109, 110]. It is emerging as an AI framework for retrieving facts from external sources. [111] examines the augmentation of context retrieval through the incorporation of external information. It proposes a sophisticated operation: the direct concatenation of pertinent information obtained from an external source to the prompt, which is subsequently treated as foundational knowledge for input into the expansive language model. Additionally, the paper introduces auto-regressive techniques for both retrieval and decoding, facilitating a more nuanced approach to information retrieval and fusion. [111] demonstrates that in-context retrieval-augmented language models, when constructed upon readily available general-purpose retrievers, yield significant LLM enhancements across a variety of model dimensions and diverse corpora. In another research, [112] showed that GPT-3 can reduce hallucinations by studying various implementations of the retrieval augmentation concept, such as Retrieval Augmented Generation (RAG) [113], Fusion-in-Decoder (FiD) [114], Seq2seq [115–117] and others. [118] developed the Chain-of-Verification (CoVe) approach to reduce hallucinations, based on letting the LLM deliberate on its own responses before self-correcting them. They suspect that extending this approach with retrieval augmentation would likely bring further gains. UNIWEB [119] converting knowledge-intensive tasks into a unified text-to-text framework and treating the web as a general source of knowledge.

3.12 Reasoning and active interaction

This subsection explores two advanced techniques that enhance the capabilities of LLMs by integrating reasoning with interaction through external tools or other action abilities. Automatic Reasoning and Tool Usage (ART) combines CoT prompting with the use of specialized tools. By guiding LLMs through multi-step reasoning and incorporating resources such as calculators and databases, ART improves the logical coherence and accuracy of model outputs. The ReAct Framework (Reasoning and Acting) synergizes reasoning with actionable steps. It prompts LLMs to devise logical sequences and interact dynamically with external tools, enabling them to handle complex, multi-step tasks efficiently. Both ART and ReAct represent significant advancements in prompt engineering, enhancing the range and reliability of tasks that LLMs can perform through the integration of reasoning and interaction.

3.12.1 Automatic reasoning and tool usage

ART is an advanced prompting technique that combines the principles of automatic CoT prompting with the strategic utilization of external tools. This method aims to enhance the reasoning capabilities of LLMs by guiding them through multi-step reasoning processes and leveraging specialized tools to achieve more accurate and relevant outputs [120].

ART builds on the CoT prompting technique, which encourages models to generate intermediate reasoning steps before arriving at a final answer. In ART, these reasoning steps are augmented by incorporating external tools such as calculators, databases, or other software applications. The integration of tools helps LLMs to perform tasks that require precise calculations, access to updated information, or specialized data processing that the model alone may not handle effectively.

For example, a prompt designed using ART might guide an LLM to first outline the steps required to solve a complex mathematical problem and then use a calculator tool to perform the necessary calculations. This combination of reasoning and tool usage ensures that the model's outputs are both logically coherent and computationally accurate.

[121] have demonstrated that ART can help models navigate complex problem spaces more effectively by breaking down tasks into manageable steps and utilizing appropriate tools at each stage. For instance, the integration of ART in natural language processing tasks has shown promising results in areas such as automated customer service, where models need to access and process information dynamically [122].

Moreover, ART's approach aligns with ongoing efforts to develop more robust and versatile AI systems capable of handling real-world tasks that demand a combination of cognitive and computational skills. [123] explores advanced ART techniques to achieve better accuracy and reliability in AI applications. These findings underscore the importance of ART in enhancing the functionality and performance of LLMs, making them more adept at handling a broader range of tasks, in particular technical problem-solving tasks that require specific and precise outputs such as financial calculations or data analysis.

3.12.2 ReAct framework

The ReAct Framework, which stands for Reasoning and Acting, synergizes the processes of reasoning and action to enable LLMs to not only think through problems but also interact with external tools and environments to achieve more accurate and contextually appropriate outcomes.

The ReAct Framework operates by prompting LLMs to generate both reasoning traces and task-specific actions. This dual approach ensures that the model first contemplates the problem, devises a logical sequence of thoughts, and then executes actions that may involve querying external databases, using calculators, or interacting with other software tools. This method is particularly effective in scenarios requiring detailed reasoning followed by specific actions, thus ensuring the LLM can handle complex, multi-step tasks efficiently [124].

For example, in a task involving financial analysis, the ReAct framework would first prompt the LLM to outline the necessary steps to evaluate a portfolio. Subsequently, the model could use financial analysis tools to gather current market data and perform calculations, integrating these results into the final analysis. This combination of reasoning and action leads to more robust and reliable outcomes compared to using static prompts alone. Another concrete example is shown in Figure 16.

By integrating reasoning and action, ReAct enables LLMs to make more informed and accurate decisions, in particular in fields such as finance, healthcare, and legal analysis, where decisions must be based on comprehensive data and logical reasoning [125]. Additionally, the framework's ability to interact with external tools ensures that the information used in decision-making is up-to-date and relevant, reducing the likelihood of errors due to outdated or incomplete data [126].

Implementing the ReAct framework is not a trivial task, as it involves developing prompts that guide LLMs through both thought processes and actions. This requires a detailed understanding of the task at hand and the tools available, ensuring that the model can seamlessly transition from reasoning to action.

4 Methodologies for multimodal large models

In recent years, vision-language models (VLMs) have made significant advancements in multimodal learning by combining visual and linguistic information. These models have demonstrated strong capabilities in tasks such as image description and visual question answering (VQA) [127–130]. While this review primarily focuses on the potential of prompt engineering in LLMs, it is also pertinent to briefly introduce

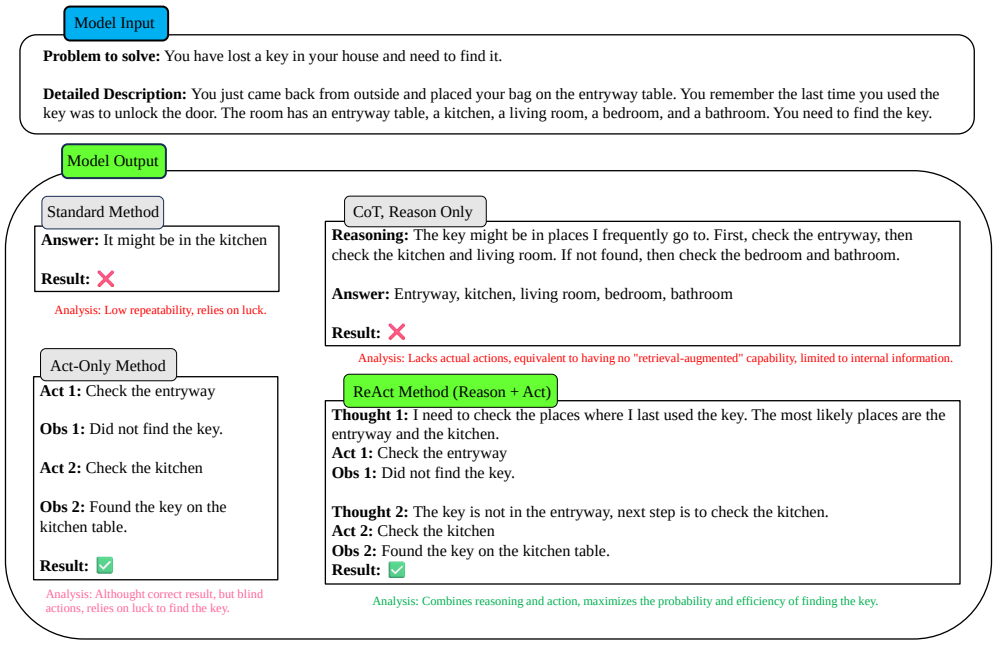

Fig. 16 An example of ReAct method.

the importance of VLMs and their applications in multimodal tasks to provide a more comprehensive perspective.

VLMs are based on the transformer architecture, and are trained on extensive datasets to learn complex semantic relationships. However, unlike early unimodal models, VLMs process both textual and visual information, enabling them to establish connections between image understanding and text generation. As can be expected, this multimodal integration makes VLMs particularly effective at handling complex tasks that involve both images and text.

To seamlessly integrate and interpret these diverse data types, VLMs require sophisticated prompt designs that ensure contextual coherence and accuracy [131, 132]. Challenges such as data alignment, modality integration, and context preservation are addressed through advanced techniques such as Context Optimization (CoOp, subsection 4.3) and Multimodal Prompt Learning (MaPLe, subsection 4.5). These advanced prompt engineering techniques enhance the ability of VLMs to generate nuanced and contextually rich outputs, thereby facilitating their effective utilization in various applications and enabling them to tackle more complex tasks [131].

4.1 Zero-shot and few-shot prompting

Zero-shot and few-shot prompting, which have already been discussed in subsection 2.7 in the context of LLMs, are also pivotal techniques in the realm of VLMs, enabling these models to handle tasks with minimal or no task-specific training data. Zero-shot prompting allows models to perform tasks without any specific examples provided during training, relying entirely on their pre-trained knowledge to generalize across new tasks and domains. For example, a model such as CLIP can be prompted with a textual description to classify images into categories it has never explicitly been trained on [3]. On the other hand, few-shot prompting involves providing the model with a small number of examples during inference, significantly enhancing the model's ability to generalize with limited data [16].

In relation to these methods, [97] systematically explored a range of prompting techniques for zero-shot and few-shot visual question answering (VQA) in vision-language models (VLMs), highlighting the impact of question templates, the integration of image captions, and the application of chain-of-thought reasoning on model performance. [16] showed the application of these techniques in CLIP,

highlighting the model's ability to generalize across different domains. Additionally, [133] presented a method for adapting CLIP to few-shot classification tasks without additional training, emphasizing practical benefits in real-world applications.

4.2 Continuous prompt vectors

Advancements in prompt engineering have enabled more effective adaptation of pre-trained VLMs to a wide range of downstream tasks. A promising approach in this domain is the use of continuous prompt vectors to fine-tune models such as CLIP for complex video understanding tasks. Unlike traditional handcrafted prompts, which require expert knowledge and manual effort, continuous prompt vectors [134] are learned during the training process, allowing for more flexible and efficient model adaptation. This method involves appending or prepending sequences of random vectors to the input text, which the model then interprets as part of its textual input. These vectors are optimized to effectively bridge the gap between the static image-based pre-training objectives and the dynamic requirements of video tasks, such as action recognition, action localization, and text-video retrieval. Additionally, lightweight temporal modeling using Transformers is applied to capture the temporal dependencies inherent in video data.

The efficiency of this approach lies in its minimal computational requirements; only a few parameters are trained, while the core model remains frozen. Despite this, the method has demonstrated competitive performance across various benchmarks, highlighting its potential in extending the capabilities of VLMs to handle resource-intensive video tasks with greater flexibility and accuracy. This continuous prompt-based adaptation represents a significant step forward in the evolution of prompt engineering, offering a scalable and effective solution for leveraging pre-trained models in more complex and diverse applications [134].

4.3 Context optimization

Context Optimization (CoOp) [135] is an innovative prompt learning approach specifically designed for VLMs. CoOp focuses on enhancing the adaptability and performance of these models by optimizing context-specific prompts. This methodology involves the creation of learnable context vectors that are embedded within the model's architecture, enabling it to dynamically adjust to different downstream tasks.

CoOp leverages the dual-stream architecture of VLMs, such as CLIP [16] and ALIGN [136], by performing context optimization on top of these pre-trained models. CoOp introduces learnable context vectors that are fine-tuned to minimize classification loss, thus avoiding extensive manual prompt engineering. By utilizing learnable context vectors, CoOp fine-tunes the prompts to align with the specific characteristics of the complex input data. This results in improved performance and better generalization across various scenarios [137]. This method is particularly valuable in applications such as image recognition and VQA, where the context can vary significantly [138].

To illustrate the practical application of CoOp, consider a VQA task [127–130]. In a VQA scenario, the model is presented with an image and a corresponding question, and it must generate an accurate answer based on the visual and textual information. By leveraging CoOp, the model utilizes learnable context vectors to optimize the prompts specific to the context of the input image and question. This process enhances the model's ability to interpret the visual elements and comprehend the textual query, leading to more precise and contextually relevant answers. For instance, if the model is shown an image of a beach scene with the question "What activity are the people engaged in?", CoOp would utilize learnable context vectors to optimize the textual prompts. These context vectors help the text encoder generate features that focus on relevant aspects of the image, such as identifying people, recognizing activities, and understanding the overall context of the scene. By aligning these optimized text features with the image features extracted by the image encoder, CoOp enables the model to generate a precise and contextually relevant answer, such as "The people are playing volleyball on the beach."

Regarding CoOp's effectiveness, [135] showed that models using CoOp significantly outperform traditional models in tasks such as image recognition and VQA. Additionally, [138] highlighted the benefits of ensembling context optimization, which further enhances the model's performance by combining multiple context vectors. This approach has been shown to improve the robustness and generalization of VLMs in real-world applications [139].

4.4 Conditional prompt learning

Conditional Context Optimization (CoCoOp) [140] is a methodology that dynamically tailors prompts based on specific conditions or contexts. Specifically, CoCoOp employs a lightweight neural network to generate input-conditional prompt vectors for each image, ensuring that the pre-trained model parameters remain unchanged. By leveraging contextual information, CoCoOp can provide more precise and relevant guidance to the model, which is particularly useful in complex, multimodal scenarios where the interplay between different types of data must be carefully managed.

One significant advantage of CoCoOp is its ability to adapt to new and unseen data without the need for fine-tuning the pre-trained model, thanks to the context-specific prompts generated by the lightweight neural network. In other words, a VLM enhanced with conditional prompts can more accurately interpret and respond to images and questions it has not encountered during training [140, 141]. This capability is critical for applications such as image captioning, VQA, and scene understanding, where the context can vary widely.

Consider an image captioning task where the goal is to generate descriptive captions for images. Using CoCoOp, the model enhances its performance with dynamically generated prompts tailored for different types of scenes. Specifically, CoCoOp extends the CoOp method by training a lightweight neural network to generate input-conditional tokens for each image. As detailed in [140], this allows the model to adapt to various contexts without extensive retraining, resulting in more accurate and contextually relevant captions. For example, a prompt for an outdoor scene might include contextual cues related to nature, weather, and activities, while a prompt for an indoor scene might focus on objects, people, and interactions. For an image of a bustling market, the conditional prompt could include cues such as "Identify the types of products being sold" or "Describe the interactions between vendors and customers". This enables the model to produce a caption such as "Vendors selling fresh fruits and vegetables in a crowded market, with customers browsing and purchasing items" [140].

This dynamic adaptation improves caption accuracy and enhances the model's ability to generalize to novel scenes, addressing the limitations of static prompt methods such as CoOp. Besides image captioning, the improved generalization capabilities of this technique make the model more robust in tasks such as VQA, image classification, and other real-world applications [142].

4.5 Multimodal prompt learning

The core idea of Multimodal Prompt Learning (MaPLe) is to introduce and optimize prompts for both the vision and language components simultaneously. By embedding prompts at various stages within the transformer architecture, MaPLe ensures that the model can adaptively learn contextual information pertinent to the specific task at hand [143]. This hierarchical approach allows the model to progressively refine its understanding and integration of multimodal inputs, leading to improved performance across a range of applications.

One of the critical innovations of MaPLe is its ability to enhance task relevance. Traditional prompt engineering often focuses on either vision or language prompts in isolation, which can limit the model's ability to fully leverage the complementary information available in multimodal data. MaPLe overcomes this limitation by jointly optimizing prompts for both modalities, thereby facilitating a more integrated and coherent representation of the input data [141, 143]. A detailed comparison between the MaPLe and traditional method (CoOp and CoCoOp) is shown in Figure 17.

Characteristic \ Method	CoOp (context optimization)	CoCoOp (Conditional Context Optimization)	MaPLe (Multi-modal Prompt Learning)
arXiv Submission Date	Sep 2021	Mar 2022	Oct 2022
Prompt Coupling	None	None	Yes (Coupling between Vision and Language prompts)
Adaptability	Limited to seen classes and specific tasks	Improved adaptability to unseen classes and various tasks	High adaptability across various tasks and unseen classes
Fine-tuning CLIP Parameters during Training	No	No	No
Performance on New Classes	Poor	Improved	Excellent
Handling of Multi-modal Data	Less effective, as it focuses on language prompts	Effective, as it integrates both image and language prompts	Highly effective, as it integrates both vision and language prompts
Computational Complexity	Moderate	Increased, due to dynamic prompt generation	High, due to multi-modal and multi-level prompt learning
Advantages	Simplifies prompt engineering; performs well on seen classes	Dynamic prompts enhance generalization to unseen classes; performs well across tasks and datasets	Multi-modal prompt learning and coupling enhance model collaboration and generalization
Disadvantages	Static prompts perform poorly on unseen classes, limited generalization; less adaptive to different tasks and datasets.	Increased computational complexity, potentially requiring more computational resources.	More complex implementation, may require more computational resources and training time.

Fig. 17 Comparison between the MaPLe and traditional method (CoOp and CoCoOp).

Another important mechanism of MaPLe, the hierarchical learning mechanism, allows the model to process and integrate information at multiple levels of abstraction. This is particularly beneficial for complex tasks that require a deep understanding of both visual and textual elements. By optimizing prompts at different layers within the transformer, MaPLe can better capture the intricate dependencies between vision and language inputs [143, 144].

[143] showed that MaPLe significantly outperforms baseline models in tasks such as image recognition and VQA. Similarly, [141] highlighted the importance of Multimodal prompt learning in enhancing the adaptability and generalization of VLMs.

To illustrate the practical application of MaPLe, consider the task of VQA [127–130]. In a typical VQA scenario, a model is provided with an image and a related question, and it must generate a correct and contextually relevant answer. Using MaPLe, the model can be fine-tuned with multimodal prompts that simultaneously address both the visual content and the textual question. For instance, given an image of a bustling market and the question “What fruit is the vendor selling?”, MaPLe would embed prompts at various levels of the transformer’s vision and language branches. These prompts might include visual prompts that focus on identifying objects and text prompts that guide the model to look for specific answer-relevant details. By processing these prompts hierarchically, the model can effectively integrate visual cues (like recognizing apples and oranges in the image) with the textual context (understanding the question) to generate an accurate answer (e.g., “The vendor is selling apples and oranges”). This multimodal approach ensures that the model leverages both the visual and textual information in a coherent and integrated manner, resulting in improved performance on VQA tasks compared to models that do not utilize such comprehensive prompt learning strategies.

5 Assessing the efficacy of prompt methods

There exist several ways to evaluate the quality of the output of an LLM. Evaluation methods can generally be divided into subjective and objective categories to assess the efficacy of current prompt methods in AIGC tools

5.1 Subjective and objective evaluations

The task of prompt engineering can be challenging because it is difficult to determine how a prompt is more effective solely based on its raw text form [145]. Therefore, evaluating prompts requires a combination of subjective and objective methods. Subjective evaluations primarily rely on human evaluators to assess the quality of the

generated content. Objective evaluations, also known as automatic evaluation methods, use algorithms to score the quality of text generated by LLMs or test on various benchmarks to quantitatively measure the efficacy of prompt methods.

Subjective evaluation and objective evaluation methods each have their advantages and disadvantages. Subjective evaluation is more in line with human intuition, but it is also more expensive and time-consuming [146]. Objective evaluation is less expensive and quicker than subjective evaluation. For instance, despite numerous pieces of research highlighting the limited correlation between BLEU and alternative metrics based on human assessments, their popularity has remained unaltered [147, 148]. The best way to evaluate the quality of LLM output depends on the specific application [149].

5.1.1 Subjective evaluations

Subjective evaluations depend on human evaluators to judge the quality of the generated content. Human evaluators can read the text generated by LLMs and score it for quality. Subjective evaluations typically include aspects such as fluency, accuracy, novelty, and relevance [34]. [150] builds a human evaluation for their "Chain of Density" (CoD) method based on "good summary" standard [151]. The four writers of the paper scored 100 summaries which include randomly shuffled CoD summaries to evaluate the performance. [77] using human judgments to compare outputs from other methods and "tree-of-thought" by asking the model to finish creative writing. They averaged the score for each output and found that the score from human judgment was consistent, which means that the results from human judges are credible. [152] invites 3 human annotators to create a set to explore the alignment between human and automatic evaluation. [146] assesses the quality with three human judges who indicated whether the generated norms and moral actions were relevant to the given moral story. Above these, subjective evaluations are increasingly used to assess content generated by models in areas that are difficult to represent with datasets and are more abstract, such as writing and summary.

5.1.2 Objective evaluations

Objective evaluations, also known as automatic evaluation methods, use algorithms to assess the quality of content generated by LLMs or to conduct tests on various benchmarks, quantitatively measuring the effectiveness of different prompt methods. Human-AI Language-based Interaction Evaluation (HALIE) [153], components of human-LM interactive systems and evaluation metrics, putting interaction at the center of LM evaluation. One kind of objective evaluation employs automated metrics, such as BiLingual Evaluation Understudy (BLEU) [154], which assigns a score to system-generated outputs, offering a convenient and rapid way to compare various systems and monitor their advancements. Other evaluations such as Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [155], and Metric for Evaluation of Translation with Explicit ORdering (METEOR) [156], assess the similarity between the generated text and reference text. More recent evaluation methods, such as BERTScore [157], aim to assess at a higher semantic level.

However, these automated metrics often fail to capture the assessment results of human evaluators fully and therefore must be used with caution [158]. So many researchers evaluate their methods by quantitating the performance of the model under specific tasks. Some of the tasks are traditional games, such as Game of 24 and 5x5 Crosswords [77]. The other tasks, in other words, called benchmarks, are datasets that contain instructions for models to finish. Exclude the comprehensive set of benchmarks such as Beyond the Imitation Game benchmark (BIGbench) [159] and Big-Bench Hard (BBH) [160], which evaluates the logical soundness of arguments, there are four kinds of benchmarks concluded below. These benchmarks provide standardized tasks and datasets that facilitate consistent and comparable assessments of different approaches. For testing prompt engineering methods, it is not to pursue the "best" benchmark but to choose the one that is most suitable for evaluating the

model's abilities, because not a single model can perform best in all kinds of tasks [161].

Math Word Problems (MWPs)

Objective evaluations about MWP test a model's ability to understand numerical-related questions. The task is challenging because the model needs to understand relevant information from natural language text as well as perform mathematical reasoning to solve it. The complexity of MWPs can be measured along multiple axes, e.g., reasoning and linguistic complexity and world and domain knowledge. Similar to earlier benchmark MATH23K [162] and Hybrid Math Word Problems dataset (HMWP) [163], simple Variations on Arithmetic Math word Problems (SVAMP) [76] is a kind of MWP benchmark to solve elementary-level math word problems, which evaluates the performance of models by asking them to give equations and answers based on the questions in elementary school. Dolphin1878 [164] is a kind of number-word problem over 1,500 number-word problems. ARIS [165] and AllArith [166] are arithmetic word problems and MAth Word ProblemS (MAWPs) [167] present algebraic word problems to test problem-solving skills. Different from these benchmarks contain one category of field, Academia Sinica Diverse MWP Dataset (ASDiv) [168], Algebra Question Answering (AQuA) [169] and MathQA [170] including more domains than others, such as arithmetic, algebraic and domain knowledge problems. SingleEQ [171] is construed with both single-step and multi-step math problems from mixed sources. MultiArith [172] includes elementary math problems with multiple steps. MATH [173] and GSM8K [60] require models to solve complex mathematical problems, emphasizing the need for a deep understanding of mathematical concepts and reasoning. Process-supervised Reward Models (PRM) 800K [174] includes 4.5K MATH test problems, and contains about 800,000 step-level labels over 75,000 solutions.

Question Answering (QA) Tasks

QA tasks require models to return feedback due to the given question. Massive Multitask Language Understanding (MMLU) [175] is a QA benchmark designed to measure knowledge acquired during pretraining by evaluating models exclusively in zero-shot and few-shot settings. Many QA benchmarks are also related to knowledge-based tasks. Fact Extraction and VERification (FEVER) [176] focuses on fact verification, requiring models to act for claims generated by altering sentences extracted from Wikipedia. MIDTERMQA [177] focuses on the 2022 U.S. midterm elections since the knowledge cutoff of black-box LLMs is often 2021 or earlier. These benchmarks play a critical role in assessing the models' abilities to comprehend, analyze, and synthesize information from diverse sources. NarrativeQA [178] built by materials such as movies and books, with nearly 63k tokens of input in each question. The Question Answering with Long Input Text, Yes (QuALITY) [179] is a multiple-choice QA dataset containing 2k–8k tokens from English source articles. CommonsenseQA [180, 181] focuses on commonsense question answering based on ConceptNet 5.5 [182], an open multilingual graph of general knowledge. HotPotQA [183] is collected by crowdsourcing such as Wikipedia articles and AI2 Reasoning Challenge (ARC) [184] includes 14M science sentences, 787 science questions, all non-diagram, and multiple choices. GovReport [185] dataset focuses on summarizing complex government reports, testing the models' ability to distill and synthesize critical information. QA benchmarks challenge models' reasoning and use of commonsense knowledge ability.

Language Understanding Tasks

In early efforts for language understanding and inductive tasks, Text REtrieval Conference (TREC) [186] focuses on the problem of retrieving answers rather than document lists. Stanford Sentiment Treebank (SST) [187] is constructed with fully labeled parse trees, enabling a comprehensive analysis of the compositional effects of sentiment in language and named SST-2 & SST-5 based on its number of labels. Summarization tasks, as tested by datasets like SummScreenFD [188] measure the effectiveness of the

methods in catching essential information from large content. AG's News [189] is a subset of the larger AG's Corpus which is built by compiling titles and description fields from articles belonging to different categories in AG's Corpus. By pairing varied task instructions with the corresponding text, SentiEval [190] decreases the sensitivities associated with prompt design during the evaluation of different LLMs. CR [191], the sentiment of sentences mined from customer reviews, and MR [192], a movie review snippet sentiment on a five-star scale, are benchmarks that instruct models to classify sentiment from contents. "Less Likely Brainstorming" [193] is a benchmark that tests by asking the model to generate outputs that humans think are relevant but less likely to happen. Subj [192] is the benchmark including the subjectivity of sentences from movie reviews and plot summaries. SAlient Long-Tail Translation Error Detection (SALTED) [194] focuses on identifying errors in translations, emphasizing linguistic proficiency and attention to detail. These evaluations highlight the models' ability to understand and process text, making accurate predictions based on the content. Coin Flip [23] dataset assesses symbolic reasoning that asks the model to answer whether a coin still heads up after people either flip or don't flip the coin.

Multimodal Tasks

Multimodal tasks are designed to evaluate a MMLMs ability to process and integrate information from multiple sources, such as text and images. RefCOCO, RefCOCO+ [195] and RefCOCOg [196] provide referring expressions for objects in images, testing models' ability to link descriptions with visual content. These evaluations are crucial for developing models capable of cross-modal understanding and interaction, essential for applications like visual question answering and image captioning.

5.2 Comparing different prompt methods

Some models are used to evaluate the performance of other models [197, 198]. The performance scores derived from different methods serve as benchmarks for evaluating models. LLM-Eval [199] is developed to measure open-domain conversations with LLMs. This method tries to evaluate the performance of LLMs on various benchmark datasets [200] such as Dynabench [201] and demonstrate their efficiency. [124, 146, 202, 203] compare their methods of prompt engineering with previous prompt methods such as CoT, Zero-shot, Natural Instructions (NI) [204], APO [205] and APE [206] though benchmarks such as SVAMP [76], GSM8K [60], ASDiv [168], AQUA [169], MultiArith [172], SingleEQ [171] and BBH [160]. Specific benchmarks are used to test the improvements of new prompt methods over the original model. [207] chooses QuALITY, SummScreenFD and GovReport under original type and long content type to compare with other methods such as Recurrence [208–210] and Retrieval [211, 212]. [213] compared their methods with APE and MI [190] by ROUGE-1, ROUGE-2 and ROUGE-L [155]. [214] calculates the score by the approach provided from [215] and compared with ReAct. [216] received better performance than other methods under RefCOCO [195], RefCOCO+ and RefCOCOg [196].

Besides comparing different methods by score, other indicators can provide additional insights. [217] adopts prediction accuracy and proof accuracy to demonstrate the advantage of Reasoning via Planning (RAP) compared with CoT under GSM8k. [218] considered the economic cost when utilizing various prompt methods. [219] reports that their Skeleton-of-Thought (SoT) method achieves nearly twice the evaluation speed. [119] divides the evaluation into 7 domains such as "dialogue", "Slot Filling", "Open-domain QA" that more comprehensively compare the ability to solve tasks. [220] normalizes the score of "accuracy", "precision" and "recall" to compare their method Chain-of-Symbol Prompting (CoS) with CoT. [221] evaluates different methods by four parts-"human", "social", "STEM" and "other".

Subjective comparison is also used in prompt methods comparison. [222] introduces the human-rater as a metric of evaluation. [223] compares its method "Planning and Executable Actions for Reasoning over Long documents (Pearl)" with other methods such as CoT, Program-of-Thought (PoT) [224], Self-Asked [225], Toolformer [226] and

ReAct in four domains, which are “explicit plan”, “iterative prompting”, “does not rely on external tools” and ‘Long documents”. [152] combines human and automatic evaluations to assess whether the method aligns with human reasoning. [77] compares CoT with ToT by human-rated “creative writing” task.

Other studies experiment mainly on certain models or tasks and employ disparate evaluation metrics, restricting comparability across methods [96, 227]. Nevertheless, recent research proposed a general evaluation framework called InstructEval [228] that enables a comprehensive assessment of prompting techniques across multiple models and tasks. InstructEval reached the following conclusions: in few-shot settings, omitting prompts or using generic task-agnostic prompts tends to outperform other methods, with prompts having little impact on performance; in zero-shot settings, expert-written task-specific prompts can significantly boost performance, with automated prompts not outperforming simple baselines; the performance of automated prompt generation methods is inconsistent, varying across different models and task types, displaying a lack of generalization.

6 Applications improved by prompt engineering

The output enhancements provided by prompt engineering techniques make LLMs better applicable to real-world applications. This section briefly discusses applications of prompt engineering in fields such as teaching, programming, and others.

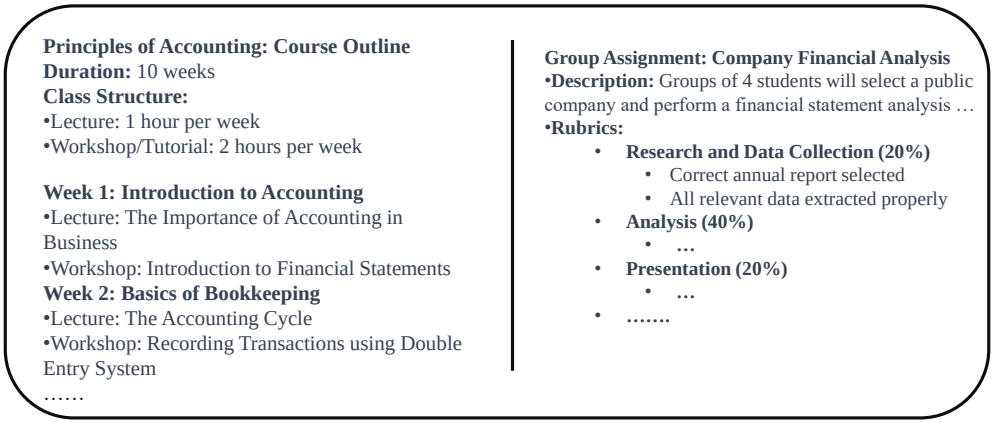

Fig. 18 Guideline of courses generated by GPT-4

6.1 Assessment in teaching and learning

[229] investigates the application of machine learning methods in young student education. In such a context, prompt engineering can facilitate the creation of personalized learning environments. By offering tailored prompts, LLMs can adapt to an individual’s learning pace and style. Such an approach can allow for personalized assessments and educational content, paving the way for a more individual-centric teaching model. Recent advancements in prompt engineering suggest that AI tools can also cater to students with specific learning needs, thus fostering inclusivity in education [230]. As a simple example, it is possible for professors to provide rubrics or guidelines for a future course with the assistance of AI. As Figure 18 shows, when GPT-4 was required to provide a rubric about a course, with a suitable prompt, it was able to respond with a specific result that may satisfy the requirement.

The advancements in prompt engineering also bring better potential for automated grading in education. With the help of sophisticated prompts, LLMs can provide preliminary assessments, reducing the workload for educators while providing instant feedback to students [231]. Similarly, these models, when coupled with

well-designed prompts, can analyze a vast amount of assessment data, thus providing valuable insights into learning patterns and informing educators about areas that require attention or improvement [232, 233].

6.2 Content creation and editing

With controllable improved input, LLMs have primarily been used in creative works, such as content creation. Pathways Language Model (PaLM) [67] and prompting approach have been used to facilitate cross-lingual short story generation. The Recursive Reprompting and Revision framework (Re^3) [234] employs zero-shot prompting [47] with GPT-3 to craft a foundational plan including elements such as settings, characters, and outlines. Subsequently, it adopts a recursive technique, dynamically prompting GPT-3 to produce extended story continuations. For another example, Detailed Outline Control (DOC) [235] aims at preserving plot coherence across extensive texts generated with the assistance of GPT-3. Unlike Re^3, DOC employs a detailed outliner and detailed controller for implementation. The detailed outliner initially dissects the overarching outline into subsections through a breadth-first method, where candidate generations for these subsections are generated, filtered, and subsequently ranked. This process is similar to the method of chain-of-though (subsection 3.1). Throughout this generation process, an OPT-based Future Discriminators for Generation (FUDGE) [236] detailed controller plays a crucial role in maintaining relevance.

6.3 Computer programming

Prompt engineering can help LLMs perform better at outputting programming codes. By using a self-debugging prompting approach [57], which contains simple feedback, unit-test, and code explanation prompts module, the text-to-SQL [237] model is able to provide a solution it can state as correct unless the maximum number of attempts has been reached. Another example, Multi-Turn Programming Benchmark (MTPB) [238], was constructed to implement a program by breaking it into multi-step natural language prompts.

Another approach is provided in [239], which introduced the Repo-Level Prompt Generator (RLPG) to dynamically retrieve relevant repository context and construct a prompt for a given task, focusing on code auto-completion tasks. The most suitable prompt is selected by a prompt proposal classifier and combined with the default context to generate the final output.

6.4 Reasoning tasks

AIGC tools have shown promising performance in reasoning tasks. Previous research has found that few-shot prompting can enhance the performance in generating accurate reasoning steps for word-based math problems in the GSM8K dataset [24, 55, 60, 67]. The strategy of including the reasoning traces in few-shot prompts [43], self-talk [240] and chain-of-thought [23], was shown to encourage the model to generate verbalized reasoning steps. [241] conducted experiments by involving prompting strategies, various fine-tuning techniques, and re-ranking methods to assess their impact on enhancing the performance of a base LLM. They found that a customized prompt significantly improved the model's ability with fine-tuning, and demonstrated a significant advantage by generating substantially fewer errors in reasoning. In another research, [47] observed that solely using zero-shot CoT prompting leads to a significant enhancement in the performance of GPT-3 and PaLM when compared to the conventional zero-shot and few-shot prompting methods. This improvement is particularly noticeable when evaluating these models on the MultiArith [242] and GSM8K [60] datasets. [243] also introduced a novel prompting approach called Diverse Verifier on Reasoning Step (DIVERSE). This approach involves using a diverse set of prompts for each question and incorporates a trained verifier with an awareness of reasoning steps. The primary aim of DIVERSE is to enhance the performance of GPT-3

on various reasoning benchmarks, including GSM8K and others. All these works show that in the application of reasoning tasks, properly customized prompts can obtain better results from the model.

6.5 Dataset generation

LLMs possess the capability of in-context learning, enabling them to be effectively prompted to generate synthetic datasets for training smaller, domain-specific models. [244] put forth three distinct prompting approaches for training data generation using GPT-3: unlabeled data annotation, training data generation, and assisted training data generation. Besides, [245] is designed for the generation of supplementary synthetic data for classification tasks. GPT-3 is utilized in conjunction with a prompt that includes real examples from an existing dataset, along with a task specification. The goal is to jointly create synthetic examples and pseudo-labels using this combination of inputs.

7 LLMs security

Prompt engineering is the process of designing and refining the inputs (prompts) given to LLMs to elicit desired and accurate responses. This technique is crucial not only for optimizing model performance but also for enhancing security. By carefully crafting prompts, researchers and developers can identify and help to mitigate vulnerabilities in LLMs. Effective prompt engineering can expose weaknesses that might be exploited through adversarial attacks, data poisoning, or other malicious activities[246]. Conversely, poorly designed prompts can inadvertently reveal or introduce security vulnerabilities in the model [246], which could then be exploited by malicious actors, leading to issues such as the disclosure of sensitive information or susceptibility to adversarial attacks.

Thus, prompt engineering serves as both a tool for improving LLMs functionality and a critical component of their security framework. The proactive, open, and in-depth efforts of researchers in identifying and mitigating vulnerabilities through prompt engineering are essential for maintaining the integrity and safety of LLMs in diverse applications [246].

This is particularly true in critical sectors such as healthcare, finance, and cybersecurity, where prompt attacks against LLMs could lead to significant breaches of sensitive information or disrupt essential services [247]. For example, adversarial attacks can manipulate model outputs to spread harmful or misleading information [3], while data poisoning during training can corrupt the model's learning process, leading to unreliable outputs. In healthcare, compromised models could lead to incorrect diagnoses and treatment plans, endangering patient lives. Similarly, in finance, compromised models could result in significant financial losses and undermine trust in automated financial services [248].

Consequently, there is a critical need for continuous and in-depth research in prompt engineering security to fully realize its benefits and address emerging challenges. A deeper understanding of attack methods and their mechanisms in relation to prompt engineering is essential for both large model developers and users to better defend against these threats. In this section, we will explore some mainstream attack methods related to prompt engineering and also discuss how to defend against them.

7.1 Adversarial attacks

Adversarial attacks involve the deliberate manipulation of input data to deceive a machine learning model into making incorrect predictions. In the context of LLMs, these attacks can take the form of subtly altered prompts or inputs that cause the model to produce unintended or harmful outputs. This manipulation exploits the sensitivity of LLMs to small perturbations in input data, revealing significant vulnerabilities [249, 250]. For instance, slight alterations in a text could mislead the model's natural language understanding, leading to incorrect or biased responses [3, 251].

The potential for adversarial attacks is particularly concerning in applications such as automated customer service or legal document analysis, where the integrity and accuracy of responses are critical [252]. One example of adversarial attacks in image recognition is illustrated in Figure 19 [253].

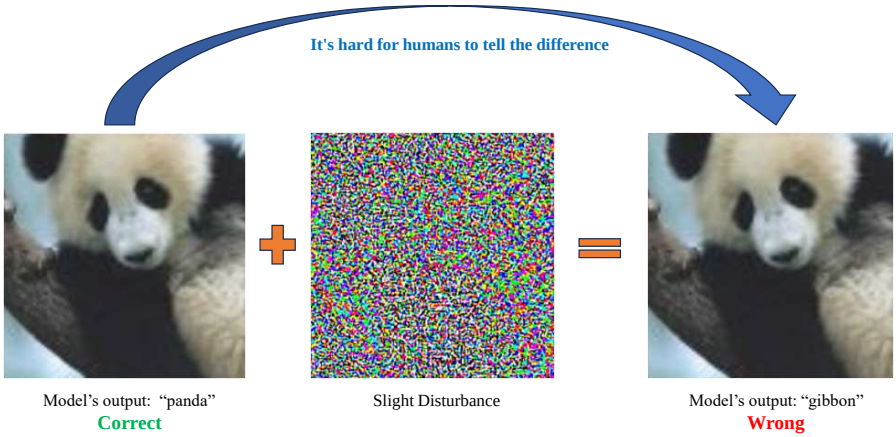

Fig. 19 An example of adversarial attack mislead the model.

Recent research has highlighted various techniques and impacts of adversarial attacks on LLMs. Adversarial demonstration attacks, for instance, can mislead models into making incorrect predictions with only subtle changes in the input data, effectively manipulating erroneous outputs across various scenarios [254]. These incorrect predictions were verified to be due to the input data changes, and not due to the inherent randomness of the models.

Optimization techniques can enhance the effectiveness of adversarial attacks to exposes various weaknesses in LLMs, making it more challenging to defend against such threats [255, 256]. For instance, in legal document analysis, adversarial inputs can lead to incorrect legal interpretations, potentially affecting case outcomes. In healthcare, such attacks could mislead models into providing incorrect medical advice, jeopardizing patient safety [257]. These examples highlight the need for effective defenses against adversarial attacks to ensure the safe and reliable deployment of LLMs for such critical applications.

7.1.1 Data poisoning

Data poisoning involves the injection of malicious data into the training set, compromising the integrity of the model. This type of attack can significantly distort the learning process, leading to erroneous outputs once the model is deployed. In LLMs, data poisoning can be especially insidious as it may go undetected during the training phase. For instance, an attacker might insert misleading or harmful data into the large corpus used to train an LLM, causing the model to learn and reproduce these inaccuracies when prompted.

This mechanism bears some similarity with how backdoor attacks [258] can be introduced in models; both involve tampering with the training data to embed malicious patterns that influence model behavior. While backdoors typically rely on specific triggers to activate unwanted behaviors, data poisoning broadly affects the model's overall performance and decision-making process. Although data poisoning is primarily associated with the training phase of model development, its implications extend to prompt engineering. Effective prompt engineering can help identify and mitigate the risks posed by poisoned data. For example, by carefully designing and testing prompts, practitioners can detect anomalies or unexpected model behaviors that may indicate underlying data poisoning. Moreover, prompt engineering can include rigorous data validation steps to ensure the training corpus is free from malicious alterations.

The implications of data poisoning are far-reaching, affecting sectors that rely on accurate data analysis and generation, such as healthcare, finance, and legal services [259]. Thus, integrating robust prompt engineering practices is crucial for preventing the inadvertent inclusion of poisoned data and safeguarding the reliability of LLMs.

7.1.2 Backdoor attacks

Backdoor threats involve embedding hidden vulnerabilities within a model that can be activated by specific prompts [260]. These backdoors, introduced during training through manipulated data, remain dormant until a trigger prompt is presented. In LLMs, a backdoor might be a specific phrase or pattern that, when encountered, triggers the model to generate a predefined, potentially harmful output, posing significant security risks due to the difficulty in detection [258, 261–266]. A visual illustration is shown in Figure 20.

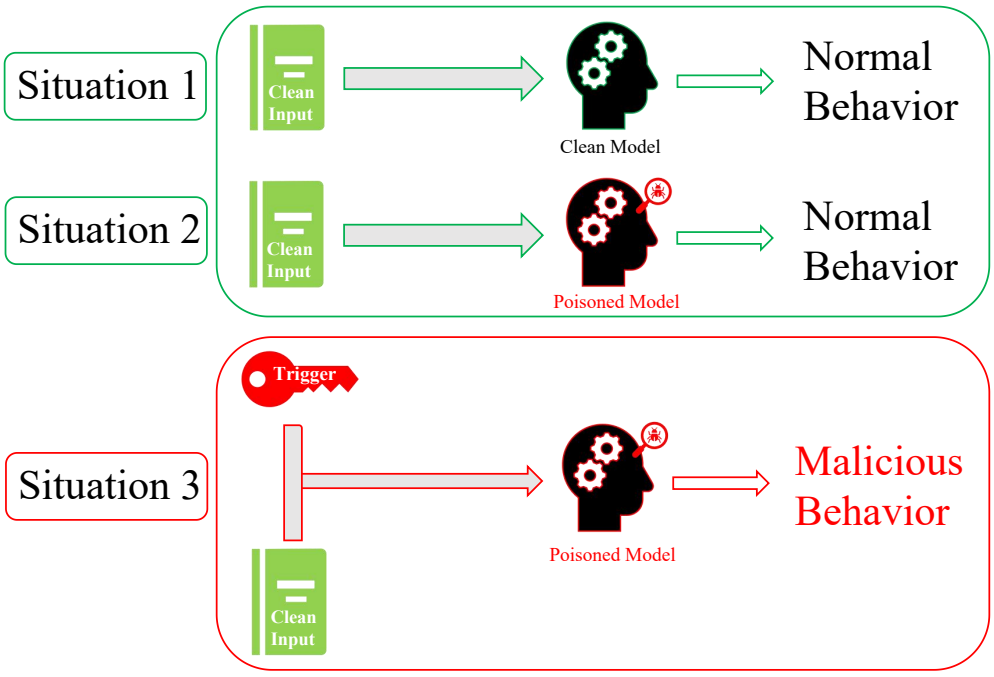

Fig. 20 An illustration of three scenarios in the backdoor attack. (1) a clean model receiving clean inputs and producing normal outputs; (2) a contaminated model receiving clean inputs but still producing normal outputs; and (3) the same model producing harmful or incorrect outputs when an implanted backdoor trigger is present.

Prompt engineering has a role to play in uncovering and mitigating backdoor threats [267]. By carefully designing and testing prompts, LLMs can be assessed for their susceptibility to adversarial attacks, data poisoning, and backdoor activations, thereby ensuring their secure deployment.

Examples of backdoor attacks include Refool [268], which leverages the natural phenomenon of reflection to stealthily implant backdoors in deep neural networks, achieving high success rates across various datasets and models while resisting state-of-the-art defenses. Similarly, [269] presents ProAttack, a clean-label backdoor method using the prompt itself as the trigger. ProAttack achieves leading attack success rates in textual backdoor attacks on LLMs, revealing critical security vulnerabilities in prompt-based learning. Building on these advancements, Imperio [270] further evolves backdoor attack strategies through language-guided instructions. By embedding backdoors during training that are later activated by natural language prompts, Imperio's approach complicates detection, as the backdoors remain concealed and can unpredictably control model behavior, even in novel scenarios.

Backdoor threats not only pose immediate security risks but also can erode trust in AI systems, emphasizing the need for transparency in AI model development. Implementing rigorous testing frameworks to detect and eliminate hidden vulnerabilities is essential before deploying models in real-world scenarios.

7.1.3 Prompt injection and prompt leaking

Poorly designed prompts can make LLMs susceptible to various types of attacks, including prompt injection and prompt leaking. Prompt injection attacks involve inserting malicious inputs into prompts to manipulate the model's output, which can result in the generation of harmful or misleading information. For example, a malicious actor could craft a prompt that subtly alters the model's response in a way that promotes false information or biases [252]. Prompt leaking, on the other hand, occurs when sensitive or proprietary information embedded in prompts is exposed, jeopardizing the security and privacy of applications that rely on LLMs [271].

7.2 Prompt hacking

Prompt hacking refers to a class of attacks that involve manipulating the input prompts provided to LLMs, with the goal of provoking unintended behaviors—ranging from benign errors to severe consequences such as misinformation dissemination or data breaches. Prompt hacking exploits the fundamental way LLMs process and generate responses. Unlike traditional hacking, which exploits software vulnerabilities, prompt hacking relies on the strategic crafting of malicious inputs to deceive the LLM into performing actions that deviate from its intended function [272–277]. This vulnerability is particularly concerning because it can be executed without the need for sophisticated technical skills. As LLMs become more integrated into various applications, the risk posed by prompt hacking increases, necessitating robust security measures to prevent such attacks [278].

[279] presents a large-scale study on the vulnerabilities of LLMs to prompt injection attacks by organizing a global prompt hacking competition, resulting in the creation of an extensive dataset and a comprehensive taxonomy of adversarial prompt types. [273] provides a comprehensive survey of security vulnerabilities in LLMs, focusing on prompt hacking and adversarial attacks, and discusses various defense mechanisms to enhance the resilience of these models against such threats.

Recognizing and addressing prompt hacking vulnerabilities is essential for developing robust LLM-based applications. By understanding prompt hacking techniques, developers can implement critical security measures such as stricter prompt validation, anomaly detection, and response filtering, which can collectively enhance the robustness of LLMs and mitigate the risks posed by malicious prompts [279]. Moreover, training programs for both developers and users as well as clear guidelines can significantly reduce the likelihood of successful prompt injection attacks, thus maintaining the integrity of LLM systems [280].

In response to these challenges, recent initiatives such as the OWASP LLM prompt hacking project [281] have emerged, offering not only valuable educational resources but also practical exercises to equip developers and security professionals with the tools to identify and prevent prompt hacking attacks, thereby reinforcing the security measures necessary to unleash the full potential of prompt engineering in LLMs. Practical security measures against prompt hacking include the use of advanced monitoring tools that detect suspicious prompt patterns and the integration of machine learning models trained to identify and block malicious prompts. Once again, these measures are crucial for ensuring the safe deployment of LLMs in sensitive applications such as healthcare, finance, and customer service [282].

7.3 Model stealing

Model stealing attacks are an adversarial misuse of prompt engineering techniques aimed at replicating the functionality or extracting proprietary knowledge from LLMs.

By crafting strategically designed prompts, attackers can systematically interact with the target model, gradually reconstructing its internal mechanics or sensitive data. This process, known as "query-based extraction", allows the attacker to build a surrogate model that mimics the target model's responses. This gradual model reconstruction approach relies on the ability to generate diverse and informative prompts that cover a wide range of inputs the model might encounter [283], and can be particularly effective when the target model is a black-box system, where the attacker has no access to the internal architecture but can observe the outputs generated in response to the inputs [284].

This stealing process highlights the vulnerabilities inherent in LLMs when exposed to malicious prompt manipulations, potentially resulting in intellectual property theft, erosion of competitive advantages, and the unethical deployment of cloned models in unauthorized contexts [285–287].

One notable example of a model stealing attack is the extraction of the projection matrix from OpenAI's language models. Researchers demonstrated how, through a series of carefully crafted prompts, they could extract significant portions of the model's architecture and parameters, effectively creating a replica of the original model [288]. Another incident involved adversaries using prompt engineering techniques to replicate commercial LLMs used in customer service, resulting in substantial intellectual property theft and financial losses for the companies involved [289].

Numerous studies have explored model stealing of LLMs. For instance, [290] proposes a novel prompt stealing attack against LLMs by introducing a two-stage approach involving parameter extraction and prompt reconstruction, effectively demonstrating the vulnerability of LLMs to reverse engineering of prompts based on their generated responses. Furthermore, [291] introduces PRSA, a novel framework for prompt stealing attacks against LLMs, which effectively infers the intent of target prompts and generates functionally equivalent surrogate prompts, highlighting the significant risks of prompt leakage in both non-interactive and interactive prompt services.

The effectiveness of these attacks underscores the need for robust defenses. Proposed countermeasures include limiting the number of queries a single user can make, implementing anomaly detection to identify suspicious querying patterns, and using defensive perturbations to mislead potential attackers [292].

7.4 Enhancing LLMs security

Adversarial example generation is a fundamental technique in AI security, designed to test and enhance the robustness of machine learning models. By creating inputs that intentionally mislead models into making incorrect predictions, researchers can identify vulnerabilities and develop strategies to mitigate them. This process is essential for ensuring that models can withstand malicious attacks and function reliably in real-world scenarios [252]. While direct research on prompt-based adversarial example generation is limited, prompt engineering remains a critical tool in testing model robustness. By designing prompts that subtly alter input data, researchers can simulate adversarial conditions and observe how models respond. For example, ambiguous or misleading prompts can reveal how susceptible a model is to producing biased or incorrect outputs, thereby identifying potential weaknesses [293].

Adversarial training, which involves training models on adversarial examples, has proven effective in enhancing model robustness [294]. [285] have shown that models exposed to a variety of adversarial inputs during training are better equipped to handle unexpected or malicious data. This method improves the resilience of models against attacks and enhances their overall reliability.

To maximize the effectiveness of adversarial training, integrating robust prompt design is essential. This involves creating prompts that not only test the model's limits but also enhance its ability to learn from adversarial conditions. Techniques such as mask filling, where portions of text are strategically manipulated, can be used to

generate adversarial examples that expose and address vulnerabilities in the model [255].

To conclude this section on security, while prompt engineering can greatly enhance the capabilities of LLMs, it can also introduce significant risks if not managed properly. The current race to release new and improved LLM functionalities cannot disregard the critical need for secure and robust design practices to combat adversarial prompts. As LLMs become further embedded in critical applications, advancing secure prompt engineering practices is essential to safeguard against misuse, minimize security vulnerabilities, and ensure safe deployment [3, 295].

8 Prospective methodologies

Beyond the advanced methodologies discussed in Section 3, several key developments on the horizon promise to substantially advance prompt engineering capabilities. This brief section discusses some noteworthy trajectories, which could shape the future of prompt engineering.

8.1 Better understanding of structures

One significant trajectory about the future of prompt engineering that emerges is the importance of better understanding the underlying structures of AI models. This understanding is crucial to effectively guide these models through prompts and to generate outputs that are more closely aligned with user intent.

At the heart of most AI models, including GPT-4, are complex mechanisms designed to understand and generate human language. The interplay of these mechanisms forms the "structure" of these models. Understanding this structure involves unraveling the many layers of neural networks, the various attention mechanisms at work, and the role of individual nodes and weights in the decision-making process of these models [296]. Deepening our understanding of these structures could lead to substantial improvements in prompt engineering. The misunderstanding of the model may cause a lack of reproducibility [297]. By understanding how specific components of the model's structure influence its outputs, one could design prompts that more effectively exploit these components.

Furthermore, a comprehensive grasp of these structures could shed light on the shortcomings of certain prompts and guide their enhancement. Frequently, the underlying causes for a prompt's inability to yield the anticipated output are intricately linked to the model's architecture. For example, [22] found evidence of limitations in previous prompt models and questioned how much these methods truly understood the model.

The exploration of AI model architectures remains a vibrant research domain, with numerous endeavors aimed at comprehending these sophisticated frameworks. A notable instance is DeepMind's "Causal Transformer" model [298], designed to explicitly delineate causal relationships within data. This represents a stride towards a more profound understanding of AI model architectures, with the potential to help people design more efficient prompts.

Along the same lines, a more comprehensive grasp of AI model architectures would also yield advancements in explainable AI. Beyond better prompt engineering, this would also foster greater trust in AI systems and promote their integration across diverse industries [299]. For example, while AI is transforming the financial sector, encompassing areas such as customer service, fraud detection, risk management, credit assessments, and high-frequency trading, several challenges, particularly those related to transparency, are emerging alongside these advancements [300, 301]. Another example is medicine, where AI's transformative potential faces similar challenges [302, 303].

8.2 Agent for AIGC tools

The concept of AI agents has emerged as a potential trajectory in AI research [304]. In this brief subsection, we explore the relationship between agents and prompt engineering and project how agents might influence the future trajectory of AI-generated content (AIGC) tools. By definition, an AI agent comprises large models, memory, active planning, and tool use. AI agents are capable of remembering and understanding a vast array of information, actively planning and strategizing, and effectively using various tools to generate optimal solutions within complex problem spaces [305].

The evolution of AI agents can be delineated into five distinct phases: models, prompt templates, chains, agents, and multi-agents. Each phase carries its specific implications for prompt engineering. Foundational models, exemplified by architectures such as GPT-4, underpin the realm of prompt engineering.

In particular, prompt templates offer an effective way of applying prompt engineering in practice [23]. By using these templates, one can create standardized prompts to guide large models, making the generated output more aligned with the desired outcome. The usage of prompt templates is a crucial step towards enabling AI agents to better understand and execute user instructions.

AI agents amalgamate these methodologies and tools into an adaptive framework. Possessing the capability to autonomously modulate their behaviors and strategies, they strive to optimize both efficiency and precision in task execution. A salient challenge for prompt engineering emerges: devising and instituting prompts that adeptly steer AI agents toward self-regulation [22].

9 Conclusion

In conclusion, prompt engineering has established itself as an essential technique for optimizing the performance of LLMs. By employing foundational methods such as clear instructions and role-prompting, alongside advanced methodologies such as chain-of-thought and self-consistency, the capabilities of LLMs can be significantly enhanced. For VLMs, innovative strategies such as CoOp and MaPLe ensure effective integration and optimization of visual and textual data. The efficacy of these methods can be rigorously assessed through both subjective and objective evaluations, confirming their impact across diverse applications, including education, content creation, and programming. Additionally, prompt engineering has a crucial role to play in fortifying LLM security, identifying vulnerabilities, and mitigating risks through adversarial training. Looking ahead, future advancements could focus on a deeper understanding of model structures and the development of AI agents, further elevating the sophistication and capability of AI systems. This comprehensive review underscores the transformative potential of prompt engineering in advancing AI capabilities, providing a structured framework for future research and applications.

10 Acknowledgement

The authors would like to acknowledgde the support from the Interdisciplinary Intelligence Super Computer Center of Beijing Normal University at Zhuhai. This work was funded by the Natural Science Foundation of China (12271047); Guangdong Provincial Key Laboratory of Interdisciplinary Research and Application for Data Science, BNU-HKBU United International College (2022B1212010006); UIC research grant (R0400001-22; UICR0400008-21; R72021114; UICR0400036-21CTL; UICR04202405-21, UICR0700041-22); Guangdong College Enhancement and Innovation Program (2021ZDZX1046).

References

[1] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17; 2017. p. 6000–6010.

[2] Bender EM, Gebru T, McMillan-Major A, Shmitchell S. On the dangers of stochastic parrots: can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency; 2021. p. 610–623.

[3] Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20; 2020. p. 1877–1901.

[4] OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, et al. GPT-4 technical report; 2024. ArXiv:2303.08774.

[5] Team G, Anil R, Borgeaud S, Alayrac JB, Yu J, Soricut R, et al. Gemini: a family of highly capable multimodal models; 2024. ArXiv:2312.11805.

[6] Google.: Google Gemini: next-generation model. Available from: https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/#sundar-note.

[7] Hulbert D.: Tree of knowledge: ToK aka Tree of Knowledge dataset for large language models LLM. Accessed: 2023-8-15. https://github.com/dave1010/tree-of-thought-prompting.

[8] Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku. Anthropic; 2024. https://paperswithcode.com/paper/the-claude-3-model-family-opus-sonnet-haiku.

[9] Anthropic.: Claude 3 model. Available from: https://www.anthropic.com/news/claude-3-family.

[10] Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, et al. Llama 2: open foundation and fine-tuned chat models; 2023. ArXiv:2307.09288.

[11] Dubey A, Jauhri A, Pandey A, Kadian A, Al-Dahle A, Letman A, et al. The Llama 3 herd of models; 2024. ArXiv2407.21783.

[12] Sarkhel R, Huang B, Lockard C, Shiralkar P. Self-training for label-efficient information extraction from semi-structured web-pages. Proceedings of the VLDB Endowment. 2023;16(11):3098–3110.

[13] Ramesh A, Pavlov M, Goh G, Gray S, Voss C, Radford A, et al. Zero-shot text-to-image generation; 2021. ArXiv:2102.12092.

[14] Marcus G, Davis E, Aaronson S. A very preliminary analysis of DALL-E 2; 2022. ArXiv:2204.13807.

[15] OpenAI.: DALL · E: creating images from text. https://openai.com/index/dall-e/.

[16] Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, et al. Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning; 2021. p. 8748–8763.

[17] Li Y, Liang F, Zhao L, Cui Y, Ouyang W, Shao J, et al. Supervision exists everywhere: a data efficient contrastive language-image pre-training paradigm. In: Tenth International Conference on Learning Representations; 2022. .

[18] OpenAI.: Hello GPT-4o. Accessed: 2024-08-04. https://openai.com/index/hello-gpt-4o/.

[19] Moore O.: Announcing GPT-4o in the API! Accessed: 2024-05-22. Available from: https://community.openai.com/t/announcing-gpt-4o-in-the-api/744700.

[20] Kaddour J, Harris J, Mozes M, Bradley H, Raileanu R, McHardy R. Challenges and applications of large language models; 2023. ArXiv:2307.10169.

[21] 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. p. 8086–8098.

[22] Webson A, Pavlick E. Do prompt-based models really understand the meaning of their prompts? In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2022. p. 2300–2344.

[23] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems. vol. 35; 2022. p. 24824–24837.

[24] Wang X, Wei J, Schuurmans D, Le QV, Chi EH, Narang S, et al. Self-consistency improves chain of thought reasoning in language models. In: Eleventh International Conference on Learning Representations; 2023. .

[25] Shanahan M, McDonell K, Reynolds L. Role-play with large language models; 2023. ArXiv:2305.16367.

[26] Maynez J, Narayan S, Bohnet B, McDonald R. On faithfulness and factuality in abstractive summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; 2020. p. 1906–1919.

[27] Bubeck S, Chandrasekaran V, Eldan R, Gehrke J, Horvitz E, Kamar E, et al. Sparks of artificial general intelligence: early experiments with GPT-4; 2023. ArXiv:2303.12712.

[28] Yong G, Jeon K, Gil D, Lee G. Prompt engineering for zero-shot and few-shot defect detection and classification using a visual-language pretrained model. Computer-Aided Civil and Infrastructure Engineering. 2022;38(11):1536–1554.

[29] Wang J, Liu Z, Zhao L, Wu Z, Ma C, Yu S, et al. Review of large vision models and visual prompt engineering. Meta-Radiology. 2023;1(3):100047.

[30] Christiano PF, Leike J, Brown T, Martic M, Legg S, Amodei D. Deep reinforcement learning from human preferences. Advances in neural information processing systems. 2017;30.

[31] Stiennon N, Ouyang L, Wu J, Ziegler D, Lowe R, Voss C, et al. Learning to summarize with human feedback. Advances in Neural Information Processing Systems. 2020;33:3008–3021.

[32] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I, et al.: Language models are unsupervised multitask learners. Assessed: 2019-02-07. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[33] Radford A, Narasimhan K, Salimans T, Sutskever I, et al. Improving language understanding by generative pre-training; 2018. Https://openai.com/research/language-unsupervised.

[34] Holtzman A, Buys J, Du L, Forbes M, Choi Y. The curious case of neural text degeneration. In: Ninth International Conference on Learning Representations; 2020. .

[35] Welleck S, Kulikov I, Roller S, Dinan E, Cho K, Weston J. Neural text generation with unlikelihood training; 2019. ArXiv:1908.04319.

[36] Xu X, Tao C, Shen T, Xu C, Xu H, Long G, et al. Re-reading improves reasoning in language models; 2023. ArXiv:2309.06275.

[37] YanSong S, JingLi Tencent A. Joint learning embeddings for Chinese words and their components via ladder structured networks. In: Proceedings of the Twenty-Seventh International Joint Conference on Artifificial Intelligence (IJCAI-18); 2018. p. 4375–4381.

[38] Luo L, Ao X, Song Y, Li J, Yang X, He Q, et al. Unsupervised neural aspect extraction with sememes. In: IJCAI; 2019. p. 5123–5129.

[39] Yang M, Qu Q, Tu W, Shen Y, Zhao Z, Chen X. Exploring human-like reading strategy for abstractive text summarization. In: Proceedings of the AAAI conference on artificial intelligence. vol. 33; 2019. p. 7362–7369.

[40] Zhang Z, Gao J, Dhaliwal RS, Jia-Jun Li T. VISAR: a human-AI argumentative writing assistant with visual programming and rapid draft prototyping; 2023. ArXiv:2304.07810.

[41] Van Buren D. Guided scenarios with simulated expert personae: a remarkable strategy to perform cognitive work; 2023. ArXiv:2306.03104.

[42] OpenAI.: Tactic: use delimiters to clearly indicate distinct parts of the input. Accessed: 2023-09-01. https://platform.openai.com/docs/guides/gpt-best-practices/tactic-use-delimiters-to-clearly-indicate-distinct-parts-of-the-input.

[43] Logan IV R, Balažević I, Wallace E, Petroni F, Singh S, Riedel S. Cutting down on prompts and parameters: simple few-shot learning with language models. In: Findings of the Association for Computational Linguistics: ACL 2022; 2022. p. 2824–2835.

[44] Shyr C, Hu Y, Harris PA, Xu H. Identifying and extracting rare disease phenotypes with large language models; 2023. ArXiv:2306.12656.

[45] Reynolds L, McDonell K. Prompt programming for large language models: beyond the few-shot paradigm. In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems; 2021. p. 1–7.

[46] 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 Comput Surv. 2023 jan;55(9).

[47] Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero-shot reasoners. Advances in Neural Information Processing Systems. 2022;35:22199–22213.

[48] Liu J, Gardner M, Cohen SB, Lapata M. Multi-step inference for reasoning over paragraphs; 2020. ArXiv:2004.02995.

[49] Ackley DH, Hinton GE, Sejnowski TJ. A learning algorithm for Boltzmann machines. Cognitive Science. 1985;9(1):147–169.

[50] Ficler J, Goldberg Y. Controlling linguistic style aspects in neural language generation. In: Proceedings of the Workshop on Stylistic Variation; 2017. p. 94–104.

[51] Xu C, Guo D, Duan N, McAuley J. RIGA at SemEval-2023 Task 2: NER enhanced with GPT-3. In: Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023); 2023. p. 331–339.

[52] Liesenfeld A, Dingemanse M. Rethinking open source generative AI: open washing and the EU AI Act. In: The 2024 ACM Conference on Fairness, Accountability, and Transparency; 2024. p. 1774–1787.

[53] Wu S, Shen EM, Badrinath C, Ma J, Lakkaraju H. Analyzing chain-of-thought prompting in large language models via gradient-based feature attributions; 2023. ArXiv:2307.13339.

[54] Zhang Z, Zhang A, Li M, Smola A. Automatic chain of thought prompting in Large language models. In: Eleventh International Conference on Learning Representations; 2023. .

[55] Lewkowycz A, Andreassen A, Dohan D, Dyer E, Michalewski H, Ramasesh V, et al. Solving quantitative reasoning problems with language models. Advances in Neural Information Processing Systems. 2022;35:3843–3857.

[56] Zhou H, Nova A, Larochelle H, Courville A, Neyshabur B, Sedghi H. Teaching algorithmic reasoning via in-context learning; 2022. ArXiv:2211.09066.

[57] Lee N, Sreenivasan K, Lee JD, Lee K, Papailiopoulos D. Teaching arithmetic to small transformers; 2023. ArXiv:2307.03381.

[58] Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Large language models perform diagnostic reasoning. In: Eleventh International Conference on Learning Representations; 2022. .

[59] Zhang H, Parkes DC. Chain-of-thought reasoning is a policy improvement operator; 2023. ArXiv:2309.08589.

[60] Cobbe K, Kosaraju V, Bavarian M, Chen M, Jun H, Kaiser L, et al. Training verifiers to solve math word problems; 2021. ArXiv:2110.14168.

[61] Huang S, Dong L, Wang W, Hao Y, Singhal S, Ma S, et al. Language is not all you need: aligning perception with language models; 2023. ArXiv:2302.14045.

[62] Shum K, Diao S, Zhang T. Automatic prompt augmentation and selection with chain-of-thought from labeled data. In: Bouamor H, Pino J, Bali K, editors. Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics; 2023. p. 12113–12139.

[63] Del M, Fishel M. True detective: a deep abductive reasoning benchmark undoable for GPT-3 and challenging for GPT-4. In: Proceedings of the 12th Joint Conference on Lexical and Computational Semantics (*SEM 2023); 2023. p. 314–322.

[64] Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C, Mishkin P, et al. Training language models to follow instructions with human feedback. Advances in Neural Information Processing Systems. 2022;35:27730–27744.

[65] Saparov A, He H. Language models are greedy reasoners: a systematic formal analysis of chain-of-thought; 2022. ArXiv:2210.01240.

[66] Tafjord O, Dalvi B, Clark P. ProofWriter: generating implications, proofs, and abductive statements over natural language. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021; 2021. p. 3621–3634.

[67] Chowdhery A, Narang S, Devlin J, Bosma M, Mishra G, Roberts A, et al. Palm: scaling language modeling with pathways; 2022. ArXiv:2204.02311.

[68] Fan A, Lewis M, Dauphin Y. Hierarchical neural story generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2018. p. 889–898.

[69] Holtzman A, Buys J, Forbes M, Bosselut A, Golub D, Choi Y. Learning to write with cooperative discriminators. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2018. p. 1638–1649.

[70] Huang J, Gu SS, Hou L, Wu Y, Wang X, Yu H, et al. Large language models can self-improve; 2022. ArXiv:2210.11610.

[71] Shum K, Diao S, Zhang T. Automatic prompt augmentation and selection with chain-of-thought from labeled data; 2023. ArXiv:2302.12822.

[72] Khalifa M, Logeswaran L, Lee M, Lee H, Wang L. Discriminator-guided multi-step reasoning with language models; 2023. ArXiv:2305.14934.

[73] Liu J, Liu A, Lu X, Welleck S, West P, Le Bras R, et al. Generated knowledge prompting for commonsense reasoning. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2022. p. 3154–3169.

[74] Zhou D, Schärli N, Hou L, Wei J, Scales N, Wang X, et al. Least-to-most prompting enables complex reasoning in Large language models. In: Eleventh International Conference on Learning Representations; 2023. .

[75] Gao L, Madaan A, Zhou S, Alon U, Liu P, Yang Y, et al. Pal: program-aided language models. In: International Conference on Machine Learning. PMLR; 2023. p. 10764–10799.

[76] Patel A, Bhattamishra S, Goyal N. Are NLP Models really able to Solve Simple Math Word Problems? In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics; 2021. p. 2080–2094.

[77] Yao S, Yu D, Zhao J, Shafran I, Griffiths TL, Cao Y, et al. Tree of thoughts: deliberate problem solving with large language models; 2023. ArXiv:2305.10601.

[78] Long J. Large language model guided tree-of-thought; 2023. ArXiv:2305.08291.

[79] Besta M, Blach N, Kubicek A, Gerstenberger R, Gianinazzi L, Gajda J, et al. Graph of thoughts: solving elaborate problems with large language models; 2023. ArXiv:2308.09687.

[80] Wang L, Ma C, Feng X, Zhang Z, Yang H, Zhang J, et al. A survey on large language model based autonomous agents; 2023. ArXiv:2308.11432.

[81] Khot T, Trivedi H, Finlayson M, Fu Y, Richardson K, Clark P, et al. Decomposed prompting: a modular approach for solving complex tasks; 2023. ArXiv:2210.02406.

[82] Diao S, Wang P, Lin Y, Zhang T. Active prompting with chain-of-thought for large language models; 2024. ArXiv:2302.12246.

[83] Li XL, Liang P. Prefix-tuning: optimizing continuous prompts for generation. In: Zong C, Xia F, Li W, Navigli R, editors. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). Online: Association for Computational Linguistics; 2021. p. 4582–4597.

[84] Sahoo P, Singh AK, Saha S, Jain V, Mondal S, Chadha A. A systematic survey of prompt engineering in large language models: techniques and applications; 2024. ArXiv:2402.07927.

[85] Settles B.: Active learning literature survey. University of Wisconsin-Madison Department of Computer Sciences.

[86] Culotta A, McCallum A. Reducing labeling effort for structured prediction tasks. In: Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2. AAAI'05. AAAI Press; 2005. p. 746–751.

[87] White J, Fu Q, Hays S, Sandborn M, Olea C, Gilbert H, et al. A prompt pattern catalog to enhance prompt engineering with ChatGPT; 2023. ArXiv:2302.11382.

[88] Desmond M, Brachman M. Exploring prompt engineering practices in the enterprise; 2024. ArXiv:2403.08950.

[89] Mondal S, Bappon SD, Roy CK. Enhancing user interaction in ChatGPT: characterizing and consolidating multiple prompts for issue resolution; 2024. ArXiv:2402.04568.

[90] White J, Hays S, Fu Q, Spencer-Smith J, Schmidt DC. ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design; 2023. ArXiv:2303.07839.

[91] Pryzant R, Iter D, Li J, Lee Y, Zhu C, Zeng M. Automatic prompt optimization with "gradient descent" and beam search. In: Bouamor H, Pino J, Bali K, editors. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics; 2023. p. 7957–7968.

[92] Chen Y, Wen Z, Fan G, Chen Z, Wu W, Liu D, et al. MAPO: boosting large language model performance with model-adaptive prompt optimization. In: Bouamor H, Pino J, Bali K, editors. Findings of the Association for Computational Linguistics: EMNLP 2023. Singapore: Association for Computational Linguistics; 2023. p. 3279–3304.

[93] Cheng J, Liu X, Zheng K, Ke P, Wang H, Dong Y, et al. Black-box prompt optimization: aligning large language models without model training. In: Ku LW, Martins A, Srikumar V, editors. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Bangkok, Thailand: Association for Computational Linguistics; 2024. p. 3201–3219.

[94] Wang X, Li C, Wang Z, Bai F, Luo H, Zhang J, et al. PromptAgent: strategic planning with language models enables expert-level prompt optimization; 2023. ArXiv:2310.16427.

[95] Bach SH, Sanh V, Yong ZX, Webson A, Raffel C, Nayak NV, et al. Promptsource: an integrated development environment and repository for natural language prompts; 2022. ArXiv:2202.01279.

[96] Deng M, Wang J, Hsieh CP, Wang Y, Guo H, Shu T, et al. RLPrompt: optimizing discrete text prompts with reinforcement learning. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing; 2022. p. 3369–3391.

[97] Awal R, Zhang L, Agrawal A. Investigating prompting techniques for zero- and few-shot visual question answering; 2024. ArXiv:2306.09996.

[98] OpenAI.: ChatGPT plugins. Accessed: 2023-10-15. https://openai.com/blog/chatgpt-plugins.

[99] OpenAI.: GPTs: introducing the latest in conversational AI. Accessed: 2024-05-22. Available from: https://openai.com/index/introducing-gpts/.

[100] Bisson S.: Microsoft build 2023: Microsoft extends its copilots with open standard plugins. Accessed: 2023-05-25. https://www.techrepublic.com/article/microsoft-extends-copilot-with-open-standard-plugins/.

[101] Ng A.: ChatGPT prompt engineering for developers. Accessed: 2023-07-18. https://www.deeplearning.ai/short-courses/chatgpt-prompt-engineering-for-developers/.

[102] Roller S, Dinan E, Goyal N, Ju D, Williamson M, Liu Y, et al. Recipes for building an open-domain chatbot. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume; 2021. p. 300–325.

[103] whatplugin.ai.: Prompt enhancer & ChatGPT plugins for AI development tools like prompt enhancer. Accessed: 2023-09-14. https://www.whatplugin.ai/plugins/prompt-enhancer.

[104] AISEO.ai.: AISEO. Accessed: 2023-8-15. https://aiseo.ai/.

[105] ChatGPT for Search Engines.: Prompt perfect plugin for ChatGPT. Accessed: 2023-10-15. https://chatonai.org/prompt-perfect-chatgpt-plugin.

[106] Prompt Perfect.: Terms of service. Accessed: 2023-09-20. https://promptperfect.xyz/static/terms.html.

[107] Lee K, Firat O, Agarwal A, Fannjiang C, Sussillo D. Hallucinations in neural machine translation; 2018. Https://openreview.net/forum?id=SkxJ-309FQ.

[108] Ji Z, Lee N, Frieske R, Yu T, Su D, Xu Y, et al. Survey of hallucination in natural language generation. ACM Computing Surveys. 2023;55(12):1–38.

[109] Lazaridou A, Gribovskaya E, Stokowiec W, Grigorev N. Internet-augmented language models through few-shot prompting for open-domain question answering; 2022. ArXiv:2203.05115.

[110] Jiang Z, Xu FF, Gao L, Sun Z, Liu Q, Dwivedi-Yu J, et al. Active retrieval augmented generation; 2023. ArXiv:2305.06983.

[111] Ram O, Levine Y, Dalmedigos I, Muhlgay D, Shashua A, Leyton-Brown K, et al. In-context retrieval-augmented language models; 2023. ArXiv:2302.00083.

[112] Shuster K, Poff S, Chen M, Kiela D, Weston J. Retrieval augmentation reduces hallucination in conversation; 2021. ArXiv:2104.07567.

[113] Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural

Information Processing Systems. 2020;33:9459–9474.

[114] Izacard G, Grave E. Leveraging passage retrieval with generative models for open domain question answering; 2020. ArXiv:2007.01282.

[115] Lewis M, Liu Y, Goyal N, Ghazvininejad M, Mohamed A, Levy O, et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; 2020. p. 7871–7880.

[116] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research. 2020;21(1):5485–5551.

[117] Roller S, Dinan E, Goyal N, Ju D, Williamson M, Liu Y, et al. Recipes for building an open-domain chatbot. In: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume; 2021. p. 300–325.

[118] Dhuliawala S, Komeili M, Xu J, Raileanu R, Li X, Celikyilmaz A, et al. Chain-of-verification reduces hallucination in large language models; 2023. ArXiv:2309.11495.

[119] Li J, Tang T, Zhao WX, Wang J, Nie JY, Wen JR. The web can be your oyster for improving Large Language Models; 2023. ArXiv:2305.10998.

[120] Paranjape B, Lundberg S, Singh S, Hajishirzi H, Zettlemoyer L, Ribeiro MT. ART: automatic multi-step reasoning and tool-use for large language models; 2023. ArXiv:2303.09014.

[121] Greyling C.: 12 prompt engineering techniques. Available from: https://cobusgreyling.medium.com/12-prompt-engineering-techniques-644481c857aa.

[122] Vinija.: Prompt engineering. Available from: https://vinija.ai/nlp/prompt-engineering/.

[123] Badhan M.: Advanced prompt engineering techniques. Accessed: 2024-05-22. Available from: https://www.mercity.ai/blog-post/advanced-prompt-engineering-techniques.

[124] Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K, et al. ReAct: synergizing reasoning and acting in language models; 2023. ArXiv:2210.03629.

[125] Li A.: ReAct: a new framework for prompt engineering in large language models. Available from: https://www.perxeive.com/blog/react-a-new-framework-for-prompt-engineering-in-large-language-models.

[126] Roberts A.: How to ReAct to simple AI agents. Available from: https://arize.com/blog-course/react-agent-llm/.

[127] Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Zitnick CL, et al. VQA: visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2015. p. 2425–2433.

[128] Wu Q, Teney D, Wang P, Shen C, Dick A, van den Hengel A. Visual question answering: a survey of methods and datasets. Computer Vision and Image Understanding. 2017;163:21–40. Language in Vision.

[129] Wang P, Wu Q, Shen C, Dick A, van den Hengel A. FVQA: fact-based visual question answering. IEEE Transactions on Pattern Analysis and Machine

Intelligence. 2018;40(10):2413–2427.

[130] Kafle K, Kanan C. Visual question answering: datasets, algorithms, and future challenges. Computer Vision and Image Understanding. 2017;163:3–20. Language in Vision.

[131] Yin S, Fu C, Zhao S, Li K, Sun X, Xu T, et al. A survey on multimodal large language models; 2024. ArXiv:2306.13549.

[132] Wu J, Gan W, Chen Z, Wan S, Yu PS. Multimodal large language models: a survey. In: 2023 IEEE International Conference on Big Data; 2023. p. 2247–2256.

[133] Zhang R, Wei Z, Fang R, Gao P, Li K, Dai J, et al. Tip-adapter: training-free adaption of CLIP for few-shot classification; 2022. ArXiv:2207.09519.

[134] Ju C, Han T, Zheng K, Zhang Y, Xie W. Prompting visual-language models for efficient video understanding. In: Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXV. Berlin, Heidelberg: Springer-Verlag; 2022. p. 105–124.

[135] Zhou K, Yang J, Loy CC, Liu Z. Learning to prompt for vision-language models. International Journal of Computer Vision. 2022 jul;130(9):2337–2348.

[136] Jia C, Yang Y, Xia Y, Chen YT, Parekh Z, Pham H, et al. Scaling up visual and vision-language representation learning with noisy text supervision. In: Meila M, Zhang T, editors. Proceedings of the 38th International Conference on Machine Learning. vol. 139 of Proceedings of Machine Learning Research. PMLR; 2021. p. 4904–4916.

[137] Ma C, Liu Y, Deng J, Xie L, Dong W, Xu C. Understanding and mitigating overfitting in prompt tuning for vision-language models; 2023. ArXiv:2211.02219.

[138] Agnolucci L, Baldrati A, Todino F, Becattini F, Bertini M, Bimbo AD. ECO: ensembling context optimization for vision-language models; 2023. ArXiv:2307.14063.

[139] Chowdhury S, Nag S, Manocha D. APoLLo : unified adapter and prompt learning for vision language models. In: Bouamor H, Pino J, Bali K, editors. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics; 2023. p. 10173–10187.

[140] Zhou K, Yang J, Loy CC, Liu Z. Conditional prompt learning for vision-language models; 2022. ArXiv:2203.05557.

[141] Ma S, Xie CW, Wei Y, Sun S, Fan J, Bao X, et al. Understanding the multi-modal prompts of the pre-trained vision-language model; 2024. ArXiv:2312.11570.

[142] Khattak MU, Wasim ST, Naseer M, Khan S, Yang MH, Khan FS. Self-regulating prompts: foundational model adaptation without forgetting; 2023. ArXiv:2307.06948.

[143] Khattak MU, Rasheed H, Maaz M, Khan S, Khan FS. MaPLe: multi-modal prompt learning; 2023. ArXiv:2210.03117.

[144] Gu J, Han Z, Chen S, Beirami A, He B, Zhang G, et al. A systematic survey of prompt engineering on vision-language foundation models; 2023. ArXiv:2307.12980.

[145] Shen L, Tan W, Zheng B, Khashabi D. Flatness-aware prompt selection improves accuracy and sample efficiency; 2023. ArXiv:2305.10713.

[146] Paul D, Ismayilzada M, Peyrard M, Borges B, Bosselut A, West R, et al. Refiner: reasoning feedback on intermediate representations; 2023. ArXiv:2304.01904.

[147] Ananthakrishnan R, Bhattacharyya P, Sasikumar M, Shah RM. Some issues in automatic evaluation of English-Hindi MT: more blues for BLEU. Icon. 2007;64.

[148] Callison-Burch C, Osborne M, Koehn P. Re-evaluating the role of BLEU in machine translation research. In: 11th Conference of the European Chapter of the Association for Computational Linguistics; 2006. p. 249–256.

[149] Stent A, Marge M, Singhai M. Evaluating evaluation methods for generation in the presence of variation. In: International Conference on Intelligent Text Processing and Computational Linguistics. Springer; 2005. p. 341–351.

[150] Adams G, Fabbri A, Ladhak F, Lehman E, Elhadad N. From sparse to dense: GPT-4 summarization with chain of density prompting; 2023. ArXiv:2309.04269.

[151] Stiennon N, Ouyang L, Wu J, Ziegler D, Lowe R, Voss C, et al. Learning to summarize with human feedback. Advances in Neural Information Processing Systems. 2020;33:3008–3021.

[152] Wang R, Wang H, Mi F, Chen Y, Xue B, Wong KF, et al. Enhancing Large Language Models Against Inductive Instructions with Dual-critique Prompting; 2023. ArXiv:2305.13733.

[153] Lee M, Srivastava M, Hardy A, Thickstun J, Durmus E, Paranjape A, et al. Evaluating human-language model interaction; 2022. ArXiv:2212.09746.

[154] Papineni K, Roukos S, Ward T, Zhu WJ. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics; 2002. p. 311–318.

[155] Chin-Yew L. ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out; 2004. p. 74–81.

[156] Banerjee S, Lavie A. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization; 2005. p. 65–72.

[157] Zhang T, Kishore V, Wu F, Weinberger KQ, Artzi Y. BERTScore: evaluating text generation with BERT. In: Ninth International Conference on Learning Representations; 2020. .

[158] Sai AB, Mohankumar AK, Khapra MM. A survey of evaluation metrics used for NLG systems. ACM Computing Surveys (CSUR). 2022;55(2):1–39.

[159] Srivastava A, Rastogi A, Rao A, Shoeb AAM, Abid A, Fisch A, et al. Beyond the imitation game: quantifying and extrapolating the capabilities of language models; 2023. ArXiv:2206.04615.

[160] Srivastava A, Rastogi A, Rao A, Shoeb AAM, Abid A, Fisch A, et al. Beyond the imitation game: quantifying and extrapolating the capabilities of language models; 2022. ArXiv:2206.04615.

[161] Chang Y, Wang X, Wang J, Wu Y, Yang L, Zhu K, et al. A survey on evaluation of large language models. ACM Transactions on Intelligent Systems and Technology. 2024;15(3):1–45.

[162] Wang Y, Liu X, Shi S. Deep neural solver for math word problems. In: Proceedings of the 2017 conference on empirical methods in natural language processing; 2017. p. 845–854.

[163] Qin J, Lin L, Liang X, Zhang R, Lin L. Semantically-aligned universal tree-structured solver for math word problems; 2020. ArXiv:2010.06823.

[164] Shi S, Wang Y, Lin CY, Liu X, Rui Y. Automatically solving number word problems by semantic parsing and reasoning. In: Proceedings of the 2015 conference on empirical methods in natural language processing; 2015. p. 1132–1142.

[165] Hosseini MJ, Hajishirzi H, Etzioni O, Kushman N. Learning to Solve Arithmetic Word Problems with Verb Categorization. In: Moschitti A, Pang B, Daelemans W, editors. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics; 2014. p. 523–533.

[166] Roy S, Roth D. Unit dependency graph and its application to arithmetic word problem solving. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31; 2017. .

[167] Koncel-Kedziorski R, Roy S, Amini A, Kushman N, Hajishirzi H. MAWPS: a math word problem repository. In: Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: human language technologies; 2016. p. 1152–1157.

[168] Miao SY, Liang CC, Su KY. A diverse corpus for evaluating and developing English math word problem solvers; 2021. ArXiv:2106.15772.

[169] Goswami M, Sanil V, Choudhry A, Srinivasan A, Udompanyawit C, Dubrawski A. AQuA: a benchmarking tool for label quality assessment; 2024. ArXiv:2306.09467.

[170] Amini A, Gabriel S, Lin P, Koncel-Kedziorski R, Choi Y, Hajishirzi H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms; 2019. ArXiv:1905.13319.

[171] Koncel-Kedziorski R, Hajishirzi H, Sabharwal A, Etzioni O, Ang SD. Parsing algebraic word problems into equations. Transactions of the Association for Computational Linguistics. 2015;3:585–597.

[172] Roy S, Roth D. Solving general arithmetic word problems; 2016. ArXiv:1608.01413.

[173] Hendrycks D, Burns C, Kadavath S, Arora A, Basart S, Tang E, et al. Measuring mathematical problem solving with the math dataset; 2021. ArXiv:2103.03874.

[174] Lightman H, Kosaraju V, Burda Y, Edwards H, Baker B, Lee T, et al. Let's verify step by step; 2023. ArXiv:2305.20050.

[175] Hendrycks D, Burns C, Basart S, Zou A, Mazeika M, Song D, et al. Measuring massive multitask language understanding; 2020. ArXiv:2009.03300.

[176] Thorne J, Vlachos A, Christodoulopoulos C, Mittal A. FEVER: a large-scale dataset for fact extraction and VERification; 2018. ArXiv:1803.05355.

[177] Feng S, Shi W, Bai Y, Balachandran V, He T, Tsvetkov Y. Knowledge card: filling LLMs' knowledge gaps with plug-in specialized Language Models; 2023. ArXiv:2305.09955.

[178] Kočiskỳ T, Schwarz J, Blunsom P, Dyer C, Hermann KM, Melis G, et al. The narrativeqa reading comprehension challenge. Transactions of the Association for Computational Linguistics. 2018;6:317–328.

[179] Pang RY, Parrish A, Joshi N, Nangia N, Phang J, Chen A, et al. QuALITY: question answering with long input texts, yes! In: North American Chapter of the Association for Computational Linguistics; 2021. p. 5336–5358.

[180] Talmor A, Herzig J, Lourie N, Berant J. Commonsenseqa: a question answering challenge targeting commonsense knowledge; 2018. ArXiv:1811.00937.

[181] Talmor A, Herzig J, Lourie N, Berant J. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 4149–4158.

[182] Speer R, Chin J, Havasi C. Conceptnet 5.5: An open multilingual graph of general knowledge. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31; 2017. .

[183] Yang Z, Qi P, Zhang S, Bengio Y, Cohen WW, Salakhutdinov R, et al. HotpotQA: a dataset for diverse, explainable multi-hop question answering. In: Conference on Empirical Methods in Natural Language Processing; 2018. p. 2369–2380.

[184] Clark P, Cowhey I, Etzioni O, Khot T, Sabharwal A, Schoenick C, et al. Think you have solved question answering? try arc, the ai2 reasoning challenge; 2018. ArXiv:1803.05457.

[185] Huang L, Cao S, Parulian N, Ji H, Wang L. Efficient attentions for long document summarization; 2021. ArXiv:2104.02112.

[186] Voorhees EM, Tice DM. Building a question answering test collection. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval; 2000. p. 200–207.

[187] Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing; 2013. p. 1631–1642.

[188] Chen M, Chu Z, Wiseman S, Gimpel K. Summscreen: a dataset for abstractive screenplay summarization; 2021. ArXiv:2104.07091.

[189] Zhang X, Zhao J, LeCun Y. Character-level convolutional networks for text classification. Advances in neural information processing systems. 2015;28.

[190] Zhang W, Deng Y, Liu B, Pan SJ, Bing L. Sentiment analysis in the era of large language models: a reality check; 2023. ArXiv:2305.15005.

[191] Hu M, Liu B. Mining and summarizing customer reviews. In: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining; 2004. p. 168–177.

[192] Pang B, Lee L. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales; 2005. Cs/0506075.

[193] Tang L, Peng Y, Wang Y, Ding Y, Durrett G, Rousseau JF. Less likely brainstorming: Using language models to generate alternative hypotheses. In: Proceedings of the conference. Association for Computational Linguistics. Meeting. vol. 2023. NIH Public Access; 2023. p. 12532.

[194] Raunak V, Post M, Menezes A. SALTED: a framework for SAlient long-tail translation error detection; 2022. ArXiv:2205.09988.

[195] Yu L, Poirson P, Yang S, Berg AC, Berg TL. Modeling context in referring expressions. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. Springer; 2016. p. 69–85.

[196] Mao J, Huang J, Toshev A, Camburu O, Yuille AL, Murphy K. Generation and comprehension of unambiguous object descriptions. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 11–20.

[197] Jain N, Saifullah K, Wen Y, Kirchenbauer J, Shu M, Saha A, et al. Bring your own data! Self-supervised evaluation for large language models; 2023. ArXiv:2306.13651.

[198] Wang Y, Yu Z, Zeng Z, Yang L, Wang C, Chen H, et al. PandaLM: an automatic evaluation benchmark for LLM instruction tuning optimization; 2023. ArXiv:2306.05087.

[199] Lin YT, Chen YN. LLM-eval: unified multi-dimensional automatic evaluation for open-domain conversations with large language models; 2023. ArXiv:2305.13711.

[200] Dehghani M, Tay Y, Gritsenko AA, Zhao Z, Houlsby N, Diaz F, et al. The benchmark lottery; 2021. ArXiv:2107.07002.

[201] Kiela D, Bartolo M, Nie Y, Kaushik D, Geiger A, Wu Z, et al. Dynabench: rethinking benchmarking in NLP; 2021. ArXiv:2104.14337.

[202] Xu W, Banburski-Fahey A, Jojic N. Reprompting: automated chain-of-thought prompt inference through gibbs sampling; 2023. ArXiv:2305.09993.

[203] Wang L, Xu W, Lan Y, Hu Z, Lan Y, Lee RKW, et al. Plan-and-solve prompting: improving zero-shot chain-of-thought reasoning by large language models; 2023. ArXiv:2305.04091.

[204] Mishra S, Khashabi D, Baral C, Hajishirzi H. Cross-task generalization via natural language crowdsourcing instructions; 2021. ArXiv:2104.08773.

[205] Pryzant R, Iter D, Li J, Lee YT, Zhu C, Zeng M. Automatic prompt optimization with "gradient descent" and beam search; 2023. ArXiv:2305.03495.

[206] Zhou Y, Muresanu AI, Han Z, Paster K, Pitis S, Chan H, et al. Large Language Models are human-level prompt engineers. In: Eleventh International Conference on Learning Representations; 2022. .

[207] Chen H, Pasunuru R, Weston J, Celikyilmaz A. Walking down the memory maze: beyond context limit through interactive reading; 2023. ArXiv:2310.05029.

[208] Chevalier A, Wettig A, Ajith A, Chen D. Adapting language models to compress contexts; 2023. ArXiv:2305.14788.

[209] Bulatov A, Kuratov Y, Kapushev Y, Burtsev MS. Scaling transformer to 1m tokens and beyond with rmt; 2023. ArXiv:2304.11062.

[210] Xu J, Szlam A, Weston J. Beyond goldfish memory: Long-term open-domain conversation; 2021. ArXiv:2107.07567.

[211] Wu Y, Rabe MN, Hutchins D, Szegedy C. Memorizing transformers; 2022. ArXiv:2203.08913.

[212] Izacard G, Grave E. Leveraging passage retrieval with generative models for open domain question answering; 2020. ArXiv:2007.01282.

[213] Guo Q, Wang R, Guo J, Li B, Song K, Tan X, et al. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers; 2023. ArXiv:2309.08532.

[214] Sridhar A, Lo R, Xu FF, Zhu H, Zhou S. Hierarchical prompting assists large language model on web navigation; 2023. ArXiv:2305.14257.

[215] Yao S, Chen H, Yang J, Narasimhan K. Webshop: towards scalable real-world web interaction with grounded language agents. Advances in Neural Information Processing Systems. 2022;35:20744–20757.

[216] Zhang C, Xiao J, Chen L, Shao J, Chen L. TreePrompt: Learning to Compose Tree Prompts for Explainable Visual Grounding; 2023. ArXiv:2305.11497.

[217] Colangeli M, Di Francesco A, Rondoni L. Finite reservoirs and irreversibility corrections to Hamiltonian systems statistics; 2023. ArXiv:2305.14922.

[218] Jiang H, Wu Q, Lin CY, Yang Y, Qiu L. Llmlingua: Compressing prompts for accelerated inference of large language models; 2023. ArXiv:2310.05736.

[219] Ning X, Lin Z, Zhou Z, Yang H, Wang Y. Skeleton-of-thought: Large language models can do parallel decoding; 2023. ArXiv:2307.15337.

[220] Hu H, Lu H, Zhang H, Song YZ, Lam W, Zhang Y. Chain-of-symbol prompting elicits planning in Large Langauge Models; 2023. ArXiv:2305.10276.

[221] Feng S, Shi W, Bai Y, Balachandran V, He T, Tsvetkov Y. Knowledge Card: Filling LLMs' Knowledge Gaps with Plug-in Specialized Language Models; 2023. ArXiv:2305.09955.

[222] Krishna S, Ma J, Slack D, Ghandeharioun A, Singh S, Lakkaraju H. Post hoc explanations of language models can improve language models. Advances in Neural Information Processing Systems. 2024;36.

[223] Sun S, Liu Y, Wang S, Zhu C, Iyyer M. Pearl: Prompting large language models to plan and execute actions over long documents; 2023. ArXiv:2305.14564.

[224] Chen W, Ma X, Wang X, Cohen WW. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks; 2022. ArXiv:2211.12588.

[225] Press O, Zhang M, Min S, Schmidt L, Smith NA, Lewis M. Measuring and narrowing the compositionality gap in language models; 2022. ArXiv:2210.03350.

[226] Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, et al. Toolformer: Language models can teach themselves to use tools. Advances in Neural Information Processing Systems. 2024;36.

[227] Zhou Y, Muresanu AI, Han Z, Paster K, Pitis S, Chan H, et al. Large Language Models Are Human-Level Prompt Engineers; 2023. ArXiv:2211.01910.

[228] Ajith A, Pan C, Xia M, Deshpande A, Narasimhan K. InstructEval: systematic evaluation of instruction selection methods; 2023. ArXiv:2307.00259.

[229] Tang J, Zhou X, Wan X, Daley M, Bai Z. ML4STEM professional development program: enriching K-12 STEM teaching with machine learning. International Journal of Artificial Intelligence in Education. 2023;33(1):185–224.

[230] Xie Q, Dai Z, Hovy E, Luong MT, Le QV. Unsupervised data augmentation for consistency training. In: Advances in neural information processing systems. vol. 33; 2020. p. 6256–6268.

[231] Ariely M, Nazaretsky T, Alexandron G. Machine learning and Hebrew NLP for automated assessment of open-ended questions in biology. International Journal of Artificial Intelligence in Education. 2023;33(1):1–34.

[232] Nilsson F, Tuvstedt J. GPT-4 as an automatic grader: the accuracy of grades set by GPT-4 on introductory programming assignments [Bachelor Thesis]. KTH Royal Institute of Technology; 2023.

[233] Schneider J, Richner R, Riser M. Towards trustworthy autograding of short, multi-lingual, multi-type answers. International Journal of Artificial Intelligence in Education. 2023;33(1):88–118.

[234] Yang K, Tian Y, Peng N, Klein D. Re3: generating longer stories with recursive reprompting and revision. In: Goldberg Y, Kozareva Z, Zhang Y, editors. Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics; 2022. p. 4393–4479.

[235] Yang K, Klein D, Peng N, Tian Y. DOC: improving long story coherence with detailed outline control. In: Rogers A, Boyd-Graber J, Okazaki N, editors. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, Canada: Association for Computational Linguistics; 2023. p. 3378–3465.

[236] Yang K, Klein D. FUDGE: controlled text generation with future discriminators. In: Toutanova K, Rumshisky A, Zettlemoyer L, Hakkani-Tur D, Beltagy I, Bethard S, et al., editors. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Online: Association for Computational Linguistics; 2021. p. 3511–3535.

[237] Elgohary A, Hosseini S, Awadallah AH. Speak to your parser: interactive text-to-SQL with natural language feedback. In: Annual Meeting of the Association for Computational Linguistics; 2020. p. 2065–2077.

[238] Nijkamp E, Pang B, Hayashi H, Tu L, Wang H, Zhou Y, et al. Codegen: an open large language model for code with multi-turn program synthesis; 2022. ArXiv:2203.13474.

[239] Shrivastava D, Larochelle H, Tarlow D. Repository-level prompt generation for large language models of code. In: International Conference on Machine

Learning; 2023. p. 31693–31715.

[240] Shwartz V, West P, Bras RL, Bhagavatula C, Choi Y. Unsupervised commonsense question answering with self-talk; 2020. ArXiv:2004.05483.

[241] Uesato J, Kushman N, Kumar R, Song F, Siegel N, Wang L, et al. Solving math word problems with process-and outcome-based feedback; 2022. ArXiv:2211.14275.

[242] Roy S, Roth D. Solving general arithmetic word problems. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing; 2015. p. 1743–1752.

[243] Li Y, Lin Z, Zhang S, Fu Q, Chen B, Lou JG, et al. Making language models better reasoners with step-aware verifier. In: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Toronto, Canada: Association for Computational Linguistics; 2023. p. 5315–5333.

[244] Ding B, Qin C, Liu L, Bing L, Joty S, Li B. Is GPT-3 a good data annotator?; 2022. ArXiv:2212.10450.

[245] Yoo KM, Park D, Kang J, Lee SW, Park W. GPT3Mix: leveraging large-scale language models for text augmentation. In: Findings of the Association for Computational Linguistics: EMNLP 2021; 2021. p. 2225–2239.

[246] Liu Y, Deng G, Xu Z, Li Y, Zheng Y, Zhang Y, et al. Jailbreaking ChatGPT via prompt engineering: an empirical study; 2024. ArXiv:2305.13860.

[247] Yao Y, Duan J, Xu K, Cai Y, Sun Z, Zhang Y. A survey on Large Language Model (LLM) security and privacy: the good, the bad, and the ugly. High-Confidence Computing. 2024 jun;4(2):100211.

[248] Rawat P.: AI at risk: OWASP top 10 critical vulnerabilities for large language models (LLMs). Available from: https://www.infosectrain.com/blog/ai-at-risk-owasp-top-10-critical-vulnerabilities-for-large-language-models-llms/.

[249] Perez F, Ribeiro I. Ignore previous prompt: attack techniques for language models; 2022. ArXiv:2211.09527.

[250] Ren K, Zheng T, Qin Z, Liu X. Adversarial attacks and defenses in deep learning. Engineering. 2020;6(3):346–360.

[251] Yin Z, Ye M, Zhang T, Du T, Zhu J, Liu H, et al. VLATTACK: multimodal adversarial attacks on vision-language tasks via pre-trained models; 2024. ArXiv:2310.04655.

[252] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples; 2015. ArXiv:1412.6572.

[253] Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L. ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition; 2009. p. 248–255.

[254] Wang J, Liu Z, Park KH, Jiang Z, Zheng Z, Wu Z, et al. Adversarial demonstration attacks on large language models; 2023. ArXiv:2305.14950.

[255] Kolter Z.: Discrete optimization for adversarial attacks on large language models. Available from: https://orc.mit.edu/events/discrete-optimization-adversarial-attacks-large-language-models.

[256] Shayegani E, Mamun MAA, Fu Y, Zaree P, Dong Y, Abu-Ghazaleh N. Survey of vulnerabilities in large language models revealed by adversarial attacks; 2023. ArXiv:2310.10844.

[257] Selvakkumar A, Pal S, Jadidi Z. Addressing adversarial machine learning attacks in smart healthcare perspectives; 2021. ArXiv:2112.08862.

[258] Bagdasaryan E, Veit A, Hua Y, Estrin D, Shmatikov V. How to backdoor federated learning. In: Chiappa S, Calandra R, editors. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics. vol. 108 of Proceedings of Machine Learning Research. PMLR; 2020. p. 2938–2948.

[259] Steinhardt J, Koh PW, Liang P. Certified defenses for data poisoning attacks; 2017. ArXiv:1706.03691.

[260] Yang H, Xiang K, Ge M, Li H, Lu R, Yu S. A comprehensive overview of backdoor attacks in large language models within communication networks. IEEE Network. 2024;p. 1–1.

[261] Chen X, Liu C, Li B, Lu K, Song D. Targeted backdoor attacks on deep learning systems using data poisoning; 2017. ArXiv:1712.05526.

[262] Shamshiri S, Sohn I. Defense method challenges against backdoor attacks in neural networks. In: 2024 International Conference on Artificial Intelligence in Information and Communication; 2024. p. 396–400.

[263] Holland R, Pal S, Pan L, Zhang LY. Backdoor attacks and generative model fairness: current trends and future research directions. In: 2024 16th International Conference on COMmunication Systems & NETworkS; 2024. p. 31–36.

[264] Khan A, Sharma I. AI-powered detection and mitigation of backdoor attacks on databases server. In: 2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things; 2024. p. 374–379.

[265] Ooi YX. Evaluation of backdoor attacks and defenses to deep neural networks [Master's thesis]. Nanyang Technological University; 2024.

[266] Li Y, Li T, Chen K, Zhang J, Liu S, Wang W, et al. BadEdit: backdooring large language models by model editing; 2024. ArXiv:2403.13355.

[267] Gu T, Dolan-Gavitt B, Garg S. BadNets: identifying vulnerabilities in the machine learning model supply chain; 2019. ArXiv:1708.06733.

[268] Liu Y, Ma X, Bailey J, Lu F. Reflection backdoor: a natural backdoor attack on deep neural networks. In: Computer Vision – ECCV 2020. Berlin, Heidelberg: Springer-Verlag; 2020. p. 182–199.

[269] Zhao S, Wen J, Luu A, Zhao J, Fu J. Prompt as triggers for backdoor attack: examining the vulnerability in language models. In: Bouamor H, Pino J, Bali K, editors. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics; 2023. p. 12303–12317.

[270] Chow KH, Wei W, Yu L. Imperio: language-guided backdoor attacks for arbitrary model control; 2024. ArXiv:2401.01085.

[271] Abdali S, Anarfi R, Barberan C, He J. Securing large language models: threats, vulnerabilities and responsible practices; 2024. ArXiv:2403.12503.

[272] Kosch T, Feger S. Risk or chance? Large language models and reproducibility in human-computer interaction research; 2024. ArXiv:2404.15782.

[273] Liu FW, Hu C. Exploring vulnerabilities and protections in large language models: a survey; 2024. ArXiv:2406.00240.

[274] Zhan Q, Liang Z, Ying Z, Kang D. InjecAgent: benchmarking indirect prompt injections in tool-integrated large language model agents; 2024. ArXiv:2403.02691.

[275] Chen K, Wang Z, Mi B, Liu W, Wang S, Ren X, et al. Machine unlearning in large language models; 2024. ArXiv:2404.16481.

[276] Wang H, Li H, Huang M, Sha L. From noise to clarity: unraveling the adversarial suffix of large language model attacks via translation of text embeddings; 2024. ArXiv:2402.16006.

[277] Gao A. Prompt engineering for large language models; 2023. SSRN:4504303.

[278] Learn prompting.: Introduction to prompt hacking. Available from: https://learnprompting.org/docs/prompt_hacking/intro.

[279] Schulhoff S, Pinto J, Khan A, Bouchard LF, Si C, Anati S, et al. Ignore this title and HackAPrompt: exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. In: Bouamor H, Pino J, Bali K, editors. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. Singapore: Association for Computational Linguistics; 2023. p. 4945–4977.

[280] Allouin A.: Understanding LLM prompt hacking and attacks. Available from: https://medium.com/@alexandre.allouin/understanding-llm-prompt-hacking-and-attacks-8781c313a25b.

[281] Karande C.: OWASP LLM prompt hacking. Available from: https://owasp.org/www-project-llm-prompt-hacking/.

[282] Teneo AI.: Navigating the challenges of prompt hacking in 2024. Available from: https://www.teneo.ai/blog/navigating-the-challenges-of-prompt-hacking-in-2024.

[283] Krishna K, Tomar GS, Parikh AP, Papernot N, Iyyer M. Thieves on sesame street! Model extraction of BERT-based APIs. In: Ninth International Conference on Learning Representations; 2020. .

[284] Papernot N, McDaniel P, Sinha A, Wellman MP. SoK: security and privacy in machine learning. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P); 2018. p. 399–414.

[285] Tramèr F, Zhang F, Juels A, Reiter MK, Ristenpart T. Stealing machine learning models via prediction APIs. In: Proceedings of the 25th USENIX Conference on Security Symposium. SEC'16. USA: USENIX Association; 2016. p. 601–618.

[286] Shen X, Qu Y, Backes M, Zhang Y. Prompt stealing attacks against text-to-image generation models; 2023. ArXiv:2302.09923.

[287] Zhang S. Defending against model extraction attacks via watermark-based method with knowledge distillation [Master's thesis]. Nanyang Technological University; 2024.

[288] Naseh A, Krishna K, Iyyer M, Houmansadr A. Stealing the decoding algorithms of language models. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. CCS '23. New York, NY, USA: Association for Computing Machinery; 2023. p. 1835–1849.

[289] Carlini N, Paleka D, Dvijotham KD, Steinke T, Hayase J, Cooper AF, et al. Stealing part of a production language model; 2024. ArXiv:2403.06634.

[290] Sha Z, Zhang Y. Prompt stealing attacks against large language models; 2024. ArXiv:2402.12959.

[291] Yang Y, Li C, Jiang Y, Chen X, Wang H, Zhang X, et al. PRSA: PRompt Stealing Attacks against large language models; 2024. ArXiv:2402.19200.

[292] Hu H, Salcic Z, Sun L, Dobbie G, Yu PS, Zhang X. Membership inference attacks on machine learning: a survey; 2022. ArXiv:2103.07853.

[293] Kurakin A, Goodfellow I, Bengio S. Adversarial machine learning at scale; 2017. ArXiv:1611.01236.

[294] Bai T, Luo J, Zhao J, Wen B, Wang Q. Recent advances in adversarial training for adversarial robustness. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization; 2021. p. 4312–4321. Survey Track.

[295] Eric M.: A complete introduction to prompt engineering for large language models. Available from: https://www.mihaileric.com/posts/a-complete-introduction-to-prompt-engineering/.

[296] Linardatos P, Papastefanopoulos V, Kotsiantis S. Explainable AI: a review of machine learning interpretability methods. Entropy. 2020;23(1):18.

[297] Recht B, Re C, Wright S, Niu F. Hogwild!: a lock-free approach to parallelizing stochastic gradient descent. In: Advances in neural information processing systems. vol. 24; 2011. .

[298] Melnychuk V, Frauen D, Feuerriegel S. Causal transformer for estimating counterfactual outcomes. In: International Conference on Machine Learning; 2022. p. 15293–15329.

[299] Novakovsky G, Dexter N, Libbrecht MW, Wasserman WW, Mostafavi S. Obtaining genetics insights from deep learning via explainable artificial intelligence. Nature Reviews Genetics. 2023;24(2):125–137.

[300] Bertucci L, Brière M, Fliche O, Mikael J, Szpruch L. Deep learning in finance: from implementation to regulation; 2022. SSRN:4080171.

[301] Maple C, Szpruch L, Epiphaniou G, Staykova K, Singh S, Penwarden W, et al. The AI revolution: opportunities and challenges for the finance sector; 2023. ArXiv:2308.16538.

[302] Amann J, Blasimme A, Vayena E, Frey D, Madai VI. Explainability for artificial intelligence in healthcare: a multidisciplinary perspective. BMC Medical Informatics and Decision Making. 2020;20(1):1–9.

[303] Rajpurkar P, Chen E, Banerjee O, Topol EJ. AI in health and medicine. Nature Medicine. 2022;28(1):31–38.

[304] Öztürk D. What does artificial intelligence mean for organizations? A systematic review of organization studies research and a way forward. The Impact of

Artificial Intelligence on Governance, Economics and Finance, Volume I. 2021;p. 265–289.

[305] Seeamber R, Badea C. If our aim is to build morality into an artificial agent, how might we begin to go about doing so? IEEE Intelligent Systems. 2023;.