A Survey on Prompt Engineering: Taxonomy and Applications

Yaoyang Liu, Zhen Zheng, Feng Zhang, Jincheng Feng, Yiyang Fu, Xiaoyong Du

Abstract—Large Language Models (LLMs) have shown exceptional performance across a diverse range of downstream tasks. Since 2022, generative AI has demonstrated significant potential and value across various application domains, including gaming, film and television, media, and finance. By 2023, the global AI-generated content (AIGC) industry has attracted over $26 billion in investment. With the increasing prevalence of LLMs, prompt engineering has emerged as a hot research topic aimed at enhancing the interactions between users and AI systems, thereby improving the overall performance of LLMs. Prompt, which serves as the input instruction for the LLM, is closely correlated with the LLM response. Prompt engineering refines the content and structure of prompts to enhance LLMs' performance without modifying the underlying model parameters. Although extensive research has been conducted on prompt engineering, a comprehensive summary is still lacking. To bridge this gap, our paper aims to investigate the existing techniques and applications of prompt engineering. In this survey, we conduct a comprehensive review of prompt engineering. Based on the principle of human problem-solving and the comprehensive function division of Agent system, we propose a novel taxonomy of prompt engineering. We categorize prompt engineering into four distinct aspects: profile and instruction, knowledge, reasoning and planning, and reliability, offering a thorough framework for understanding its multifaceted dimensions. This taxonomy helps in the systematic design of the prompt. By comprehensively summarizing the existing prompt engineering techniques, we summarize the application of LLM across various domains and their interrelation with prompt engineering strategies. This survey summarizes the current progress of prompt engineering and its important role in advancing AI applications. We aim to provide a systematic reference for future research and applications.

Index Terms—prompt engineering, large language models, ai agent, survey, taxonomy.

1 Introduction

Large Language Models (LLMs) such as GPT-4 have attracted substantial interest due to their advanced language comprehension and generation capabilities [1, 2]. Employing task-specific instructions, or prompts, effectively leverages the diverse competencies that LLMs acquire during pretraining phase [3]. Despite exhibiting extraordinary capabilities, LLMs still encounter numerous challenges in practical applications. The integration of Reinforcement Learning from Human Feedback (RLHF) training endows LLMs with human-like conversational skills. Consequently, users may be motivated to express uniquely human modes of thought directly to the models using natural language. However, this anthropomorphic assumption does not hold for LLMs [4]. As a result, the responses generated by LLMs frequently lack the necessary professionalism and precision. Due to the generalization capabilities of LLM, unstructured prompts can lead to highly random and diverse outputs. Additionally, LLMs commonly suffer from the "hallucination" problem, where the generated content includes logical fallacies, fabricated facts, and data-driven biases. Thus, it is challenging to design the content and structure of prompts to make the responses from LLMs more professional, stable, and reliable.

Significant efforts have been directed towards implementing prompt engineering techniques to support LLMs to excel across a variety of tasks and domains. Specifically, the researchers sought to enhance the efficacy of the prompt on the LLM with diverse methodologies, including role-play prompting [5, 6, 7], task instruction [8], knowledge augmentation [9, 10, 11, 12, 13] and recursive prompting for reasoning [14, 15, 16]. For instance, Jason Wei et al. [17] introduces a linear method for recursive prompting known as the "Chain-of-Thought" (CoT) approach. Given the shortcomings of CoT's greedy strategy and the linear structure, Long [18] develops an approach called the Tree-of-Thought, which utilizes a tree-based structure. Similarly, Yao et al. [19] proposes the Graph-of-Thought approach, leveraging graph structures. As the application of system-level techniques in prompt engineering grows, LangChain-ai [20] developes Langchain, a comprehensive framework that integrates prompt templates with various system components to enhance the optimization of prompts for final submission to LLMs.

There have been seveal surveys on prompt engineering. Liu et al. [21] proposes the concept of prompt-based learning and further provides an overview of the fundamental principles of prompt engineering. This paper presents a comprehensive range of prompt forms, including soft and hard prompts. Wei et al. [22] explores the performance of LLM under different prompt techniques, with emphasis on its emergence ability. This paper reviews the challenges and outlines future directions in enhancing model performance through prompt engineering. More recently, Sahoo et al. [23] offers a detailed description of the techniques and applications of prompt engineering. However, they mainly classify the methodology around application domains. In contrast, Li et al. [24] focuses on summarising goal-oriented prompt engineering techniques based on the principles of human reasoning. However, it is important to note that most of these surveys focus on summarizing prompt engineering within fixed scenarios, but do not delve into all aspects from

• *Y Liu, F Zhang, J Feng, Y Fu, X Du are with the School of Information, Renmin University of China, Beijing, China, 100872. E-mail: {yaoyangliu, fengzhang,duyong}@ruc.edu.cn.*

the perspective of principles. Some surveys concentrate on specific aspects of prompt engineering. For instance, Qiao et al. [15] explores prompt engineering techniques aimed at enhancing reasoning performance, while Yu et al. [16] discusses these techniques from the perspective of philosophical reasoning. Meanwhile, Mialon et al. [25] centers on the augmentation of prompts. Therefore, there is a lack of surveys specifically focused on the techniques and practical design of prompt engineering.

We construct a comprehensive directory of prompt engineering techniques and provide a summary of the corresponding applications of the techniques. This can be easily comprehended and readily implemented for swift experimentation by developers and researchers. To this end, we restrict our focus to discrete prefix prompts [26] rather than cloze prompts [27, 28], because modern LLM architectures that utilize prefix prompts (specifically decoder-only models), demonstrate superior performance and are widely utilized across various fields. Furthermore, our study concentrates on hard (discrete) prompts [21, 29] rather than soft (continuous) prompts [21, 30]. Finally, we refine the scope of prompt engineering as the technology to change the content of the prompt. This allows us to cover the full range of user and system-level technologies.

To the best of our knowledge, there lacks a survey on reviewing existing prompt engineering techniques from a principle perspective and on summarizing applications of prompt engineering. In contrast to previous surveys, the primary contributions of this paper are as follows.

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

The rest of the paper is organized as follows. In Section 2, we introduce the LLMs' capabilities and the concept of prompt engineering. We simultaneously explain the motivation for our taxonomy (human problem solving principles) and the focus of our paper. In Section 3, we propose four aspects to classify prompt engineering and summarize existing studies on prompt engineering. we then provide a comprehensive classification of the applications of prompt engineering in Section 4. Last, we conclude our paper in Section 5.

2 Background

Prompt engineering is the technique of guiding model output without altering parameters by strategically designing task-specific instructions (prompts). Prompt engineering enables LLMs to perform excellently across diverse tasks and fields. Research has consistently shown that the text input to LLMs significantly influences their effectiveness in downstream tasks [31, 32]. Thus, prompt engineering serves as a strategic tool to guide model outputs without modifying underlying parameters. Given the intrinsic link between prompts and LLM's functions, prompts must be designed with a deep understanding of the capabilities of LLMs.

2.1 LLMs' Capabilities

The performance of a Large Language Model (LLM) is highly dependent on its capabilities. When designing prompts for LLMs, a comprehensive understanding of their diverse capabilities enables the creation of more effective and task-specific prompts. Existing literature [33] highlights that LLMs exhibit a broad range of capabilities, including reasoning, understanding, instruction-following, and in-context learning, among others.

Reasoning is the core ability of LLMs, which can realize complex multi-step reasoning through specific prompt design methods. As a fundamental problem-solving ability, the reasoning ability of LLMs provides support for various practical applications such as medical diagnosis, legal decisions, virtual assistants, etc. Instruction following refers to the LLM completing a new task according to the task instructions without using an example. The generalization ability of LLMs makes it possible to generate, summarize, extract, classify, and rewrite text by designing instructions.

In-context learning (ICL) is a paradigm that enables LLMs to perform tasks by learning from a few examples provided within the context [34]. This approach diverges from the conventional supervised learning that requires extensive training datasets and model parameter updates. The essence of ICL lies in its ability to leverage analogy and pattern recognition. When faced with a new query, LLMs refer to the demonstration examples in the context to identify and apply relevant patterns without altering their underlying parameters. This is akin to how humans learn from a few instances and generalize the knowledge to new situations. The model's predictions are made by concatenating the query with the contextual examples and using a scoring function to determine the most likely outcome. Prompt engineering primarily leverages ICL capabilities to steer the output of LLMs by carefully designing the content and structure of the input context.

2.2 Classification

Considering the common methods and systematic generalizations of various prompt engineering techniques, we propose a new taxonomy grounded in the function division of agent system. This approach aims to ensure both the comprehensiveness and orthogonality of our taxonomy. Current research in agent systems divides agent functions into four main categories: profile function, memory function, planning function and action function [35, 36, 37]. This

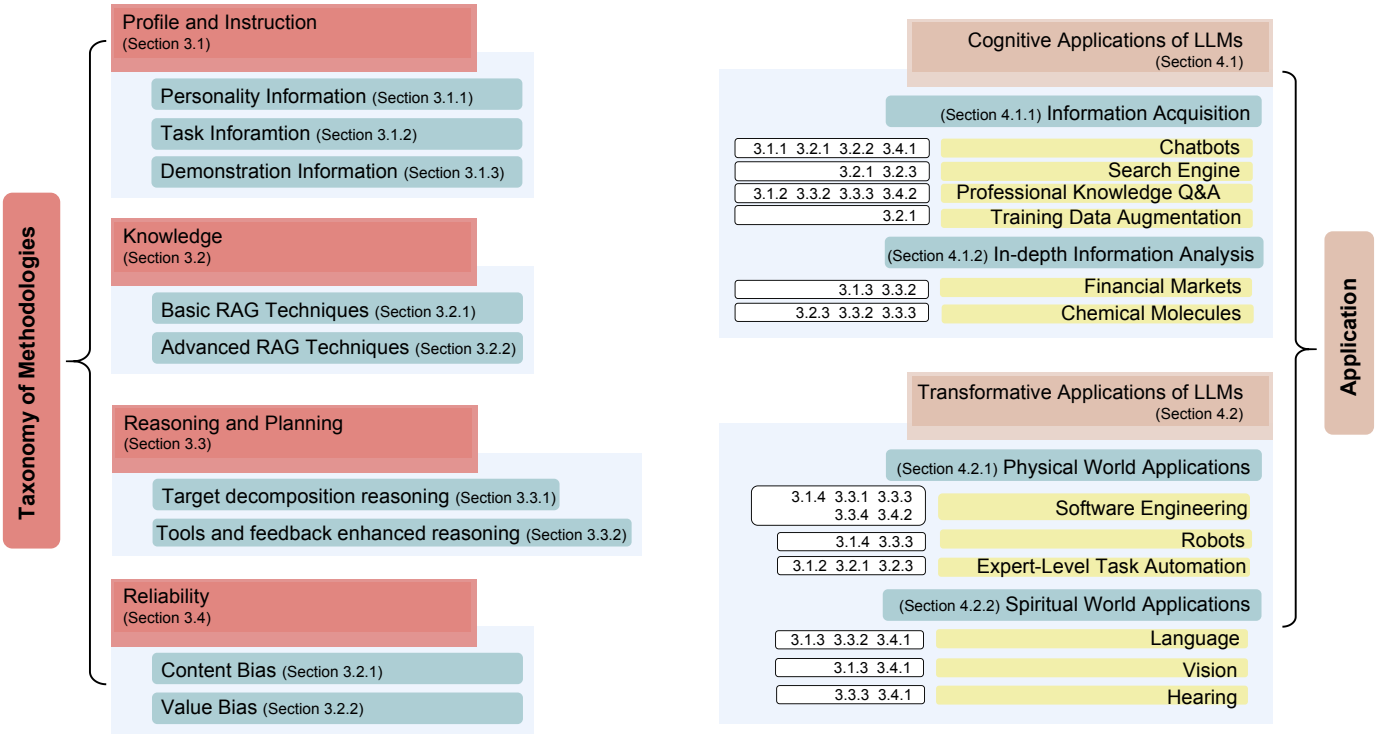


Fig. 1: The Taxonomy tree of the Prompt Engineering's methodologies. The application tree on the right reveals the relationship between the practical application of the LLM and our divided prompt engineering techniques.

comprehensive division ensures that any workflow based on LLM can be covered by these four aspects. Building on this functional division, we align prompt engineering workflows with these categories. We classify prompt engineering by four aspects:

- **Profile and Instruction**. Profile and Instruction is the basic technique for defining the basic attributes and scenarios for the work of LLM. It standardizes LLM's responses based on the information provided through natural language prompts. Profile and Instruction focuses on the design of prompt from the perspective of text content, and is the basic part of all prompt techniques.
- **Knowledge**. Knowledge is a technique that involves incorporating information from a local database into the prompt to mitigate the "hallucination" problem and improve the professionalism of the LLM.
- **Reasoning and Planning**. Reasoning and Planning is a prompt technique to enhance the reasoning ability of LLMs. This approach includes decomposing goals such as Chain of Thought as well as its derived techniques. It also includes the use of tools and feedback to facilitate reasoning. Reasoning and Planning is an important technique to improve the performance of LLMs when handling complex tasks.
- **Reliability**. Reliability refers to the process of reducing the bias of LLM's responses. This involves ensuring both the stability of content generated multiple times (content bias) and the generation of content that is free of bias, stereotypes, or cultural impairment (value bias). Reliability ensures that the LLM can steadily generate content that meets the requirements of the task without prejudice, stereotype, or cultural harm in practical application.

This taxonomy reflects the four key considerations in prompt design: setting task-related information, supplementing with external information, facilitating complex reasoning, and enhancing response quality. To ensure that LLMs execute tasks comprehensively and with high quality, it is essential to design prompts that encompass these four aspects.

2.3 Application

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

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

- **Information Acquisition.** Information Acquisition is the process of extracting pertinent knowledge from

LLMs through structured dialogues. In this process, the LLM outputs the knowledge that meets the task requirements. The use of LLM as Chatbots and Professional Knowledge Q&A are common applications of LLM as assistants to provide users with the information they need. In computer science, the information acquisition capabilities of LLMs are also applied in search engines and training data augmentation. LLMs enhance system performance by serving as advanced components within these applications.

- **In-depth Information Analysis.** This type of application leverages LLMs to analyze and reason over the user's input data. In this process, the LLM generates conclusions derived from the underlying information. In the field of Financial Markets and Chemical Molecules, LLMs perform comprehensive analyses of complex data from multiple dimensions, deriving valuable insights and summaries.

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

- **Physical World Applications.** By employing prompt engineering techniques, such as Task Information and RAG, LLMs can autonomously execute tasks in fields such as software engineering and robots. With the advancement of prompt engineering methodologies, LLMs have the potential to undertake specialized professional roles, such as AI-driven doctors and AI legal advisors.
- **Spiritual World Applications.** By leveraging prompt engineering techniques, LLMs are capable of generating literature and music. In this process, LLMs effectively replace humans in the creative workflow. With the advent of increasingly multimodal LLMs, the application of these models in video creation has emerged as a significant area of development.

3 TAXONOMY

This section presents an overview of current prompt engineering techniques in the context of generative model prompting. As shown in Figure 1, we further refine them according to the distinctive features of the different stages of the processing task.

3.1 Profile and Instruction

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

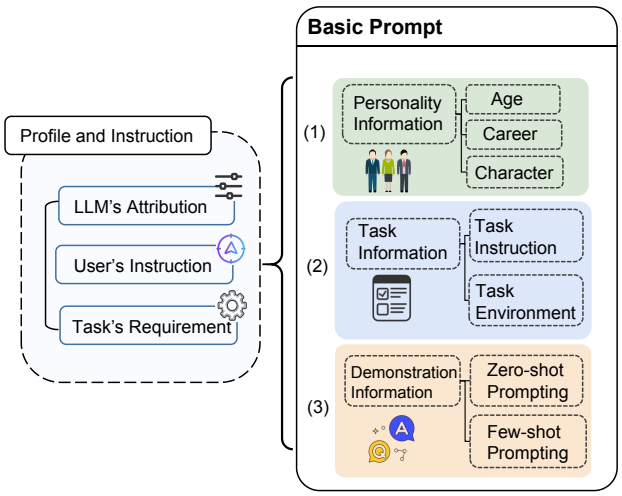


Fig. 2: Profile and Instruction technologies of prompt engineering.

The primary purpose of this line of work is to determine a reasonable knowledge location for the LLM [38], concretely embodied in personality information (Section 3.1.1), task information (Section 3.1.2) and demonstration information (Section 3.1.3), as shown in Figure 2.

3.1.1 Personality Information

LLMs can perform tasks by adopting specific roles, such as lawyers, teachers, and domain experts [5, 6]. Personality information forms a crucial part of the profile, defining the attributes of the LLM's role. These features are often included at the start of prompts to shape the LLM's response. Typically, personality information covers key characteristics like age, gender, and profession [7], along with tone and psychological traits, thereby reflecting the personality of the LLM's role. Assigning a specific role to an LLM is essentially establishing a suitable starting point for its responses. Typically, these roles are closely aligned with particular tasks. For instance, texts containing information about law are more likely to appear in the context of the word 'lawyer'. As a result, the law-related information receives more attention along with the word 'lawyer' during the inference stage, leading it to generate responses pertinent to legal matters.

3.1.2 Task Inforamtion

A. Task Instruction. Task instruction is a prompt technique used to standardize the output of LLM by incorporating clear task objectives and requirements into the prompt. This approach can be summarized into three key aspects: *intent*, *domain*, and *demand*. As shown in Figure 3, intent defines the task's goal, demand specifies the detailed requirements, and domain identifies the information source relevant to the task. Research has shown that when fundamental instructions lack clarity, LLM responses tend to be overly general [39]. If the prompt's structure is ambiguous and the content too broad, the LLM is faced with numerous options, making it difficult to focus on the critical parts of the prompt. Consequently, this can lead to extensive but unfocused results. Studies such as Avia and Levy [8] suggest that instruction understanding is a promising alternative

paradigm for few-shot learning. Compared to examples, instructions provide stronger expressiveness and more stringent constraint capabilities [8].

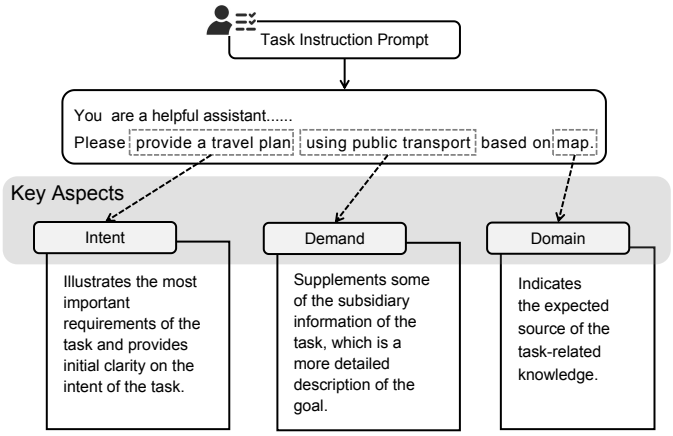


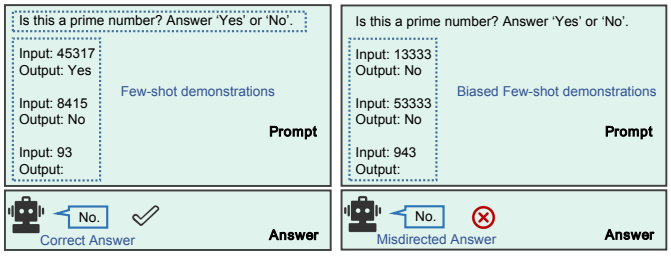
Fig. 3: An example prompt of the task instruction paradigm. The prompt consists of Intent, Demand, and Domain, providing the basic information elements of the task.

B. Task Environment. Environment Information incorporates task environment details into the prompt depending on the type of task. Providing information regarding the virtual task environment of the LLM can facilitate response generation, ensuring alignment with task requirements. For instance, if the LLM is tasked with "Fetch a bottle from the kitchen," the LLM+P model can incorporate information about the task environment and the cost of the action, such as the distance from the current location to the kitchen and the location of the objects that the task needs to interact with in the prompt. This facilitates the generation of a more feasible plan, represented using a PDDL (Planning Domain Definition Language) framework [31]. However, environmental information alone is not sufficient. The LLM should also learn the anticipated consequences of its actions and verify if the current environment meets the conditions specified in the prompt. DEPS [40] addresses this by describing the objects accessible to the LLM and defining an action format that outlines both the conditions and potential outcomes of those actions. This approach enables the LLM to understand the relationships between elements in the task environment, comprehend the environmental information, and generate responses consistent with the task requirements. These methods have shown significant improvements over traditional deep learning techniques in open-world scenarios, such as Minecraft [41].

3.1.3 Demonstration Information

Demonstration information is a technique that involves adding specific input-output mappings to the prompt. Research indicates that LLMs trained on sufficiently large and diverse datasets provide responses to zero-shot prompts that are comparable to those generated after supervised learning [42]. Based on the number of labeled demonstrations, it can be categorized into zero-shot prompting with no examples and few-shot prompting with few examples.

A. Zero-shot Prompting. Zero-shot prompting does not involve adding a labeled example to the prompt, it is composed of task and profile information [43]. This approach leverages the pre-existing knowledge of LLMs to generate responses based on the instructions of the task. Previous research [38] has shown that zero-shot prompting enables the model to access its existing knowledge by identifying already learned tasks. Furthermore, Reynolds and McDonell [38] suggest that there is significant potential for developing automated methods to generate task-appropriate zero-shot prompts.



Few-shot Prompting Misdirected Few-shot

Fig. 4: The biased demonstrations direct the LLM to the wrong location of the knowledge. This causes LLM to over-reference incomplete examples.

B. Few-shot Prompting. Few-shot prompting, unlike zero-shot prompting, provides LLMs with a few input-output examples (Figure 4) to help the model grasp the task's intention and format. Providing several high-quality examples can improve the model's performance on complex tasks and standardize the form of outputs [44]. Carefully designed demonstrations within the prompt can achieve results comparable to fine-tuning, with the performance gap narrowing as the number of model parameters increases [3]. Several approaches are proposed for selecting and augmenting these demonstrations. For example, Liu et al. [45] and Su et al. [46] enhance performance by choosing examples similar to the query input. Specifically, Liu et al. [45] employ a K-nearest neighbor (KNN) approach, while Su et al. [46] introduce Vote-K to incorporate diverse and representative examples by artificially labeling useful, previously unlabeled examples. Additionally, Jiang et al. [47] optimize the structure of examples, moving beyond the conventional "Q: A:" format to identify the most suitable prompt template for each query through large corpus analysis.

However, few-shot prompting requires additional tokens to incorporate the demonstrations, which presents a limitation for processing long text inputs. Recent research demonstrates that the selection and variation of samples can significantly impact model performance [48, 31, 49]. In response to these findings, Fei et al. [50] introduce a systematic method for measuring label biases, identifying three distinct types of label biases in in-context learning (ICL) for text classification. To address these biases, several approaches are proposed to calibrate the model's output probabilities [49, 51]. These methods typically utilize output probabilities generated from a set of inputs either sourced from the task domain [51, 52] or from standard task inputs [53]. By adjusting the samples in few-shot learning, these techniques aim to produce unbiased outputs.

3.2 Knowledge

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

Knowledge techniques involve augmenting prompts with external documents or knowledge databases. This process integrates content from local knowledge databases into the original prompt, enhancing alignment with specific information and addressing challenges such as hallucination and timeliness in LLM training [54, 55, 56]. As the scale of parameters of LLM grows, the computing resources required for fine-tuning also increase. Retrieval-Augmented Generation (RAG) is a key technology for supplementing LLMs with real-world information, enabling them to fully utilize their reasoning capabilities. Although RAG techniques encompass both retrieval and knowledge graph methodologies [54, 57], this paper primarily focuses on summarizing RAG techniques that are directly related to prompt design, while excluding those unrelated to prompt engineering. According to the complexity of the RAG framework, we classify it into basic RAG techniques and advanced RAG techniques.

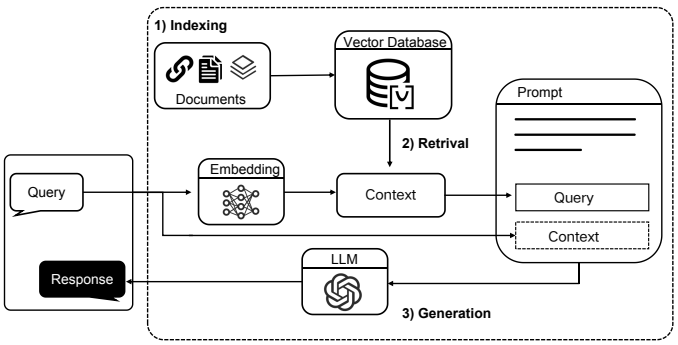


Fig. 5: A representative instance of the RAG process applied to question answering.

3.2.1 Basic RAG Techniques

Basic Retrieval-Augmented Generation (RAG) techniques dynamically retrieve information from external knowledge sources, organize the final prompt based on the original query, and use the retrieved data as a reference. The RAG workflow typically consists of three main steps [9], as illustrated in Figure 5:

(1) Indexing: Documents are divided into smaller chunks, encoded into vector representations, and stored in a database, such as an inverted index or a vector database.

(2) Retrieval: The top k chunks most relevant to the query are retrieved based on semantic similarity.

(3) Generation: The original query and the retrieved chunks are fed into a large language model (LLM) to generate the final response.

A simple RAG prompt might look like: "Please answer the above question based on: Segment 1: Segment 2:." The segment chunks are then populated with retrieved results from the external knowledge source. Basic RAG techniques provide a foundational approach to augmenting LLMs with external knowledge. This can be implemented using functions and other relevant tools. There are several potential areas for optimization within basic RAG techniques [58]. From a prompt engineering perspective, three key challenges are: 1) retrieving the most relevant documents; 2) effectively combining the retrieved content to achieve optimal results; and 3) iteratively refining the entire process [58].

3.2.2 Advanced RAG Techniques

In this section, we begin with an overview of the representative RAG framework and then explore optimization methods from three distinct perspectives: retrieval optimization, post-retrieval optimization, and memory management.

A. Advanced RAG Workflow. As illustrated in Figure 6, advanced RAG enhances the alignment between retrieved knowledge and the prompt, enabling the construction of more coherent and effective prompts. The advanced RAG prompt engineering enhances the workflow of basic RAG by incorporating several key optimizations:

(1) In the pre-retrieval stage, it addresses semantic discrepancies between the query and document chunks by rewriting and decomposing the query.

(2) In the post-retrieval stage, it refines results by modifying the content and structure of the "query+document".

(3) Additionally, advanced RAG leverages retrieval history and conversation history to generate memory for LLMs, improving the accuracy of responses in long conversations.

B. Advanced RAG Optimization techniques. This section introduces prompt engineering techniques utilized in advanced RAG frameworks. Current research focuses on optimizing three key areas: pre-retrieval, post-retrieval, and memory management.

1) Pre-retrieval Optimization. The pre-retrieval optimization focuses on augmenting the original query in order to retrieve documents that are more relevant to the task. MultiHop-RAG [10] is a newly developed dataset designed to strengthen RAG's capabilities. This dataset includes a knowledge base, multi-hop queries that facilitate reasoning across multiple documents, fundamental factual answers, and relevant supporting evidence. The queries are categorized into inference, comparison, temporal, and null queries, each tailored to specific types of reasoning tasks. The methodology for constructing the dataset involves gathering data, extracting evidence, establishing connections between entities and topics, generating queries and answers. Query standardization is performed based on their unique characteristics, ensuring consistency and effectiveness across all four query types.

Query rewriting is another technique employed in pre-retrieval optimization. Query2doc [11] and ITER-RETGEN [59] both illustrate the effectiveness of query rewriting. These approaches leverage the capabilities of LLMs to generate pseudo-documents, which are then combined with

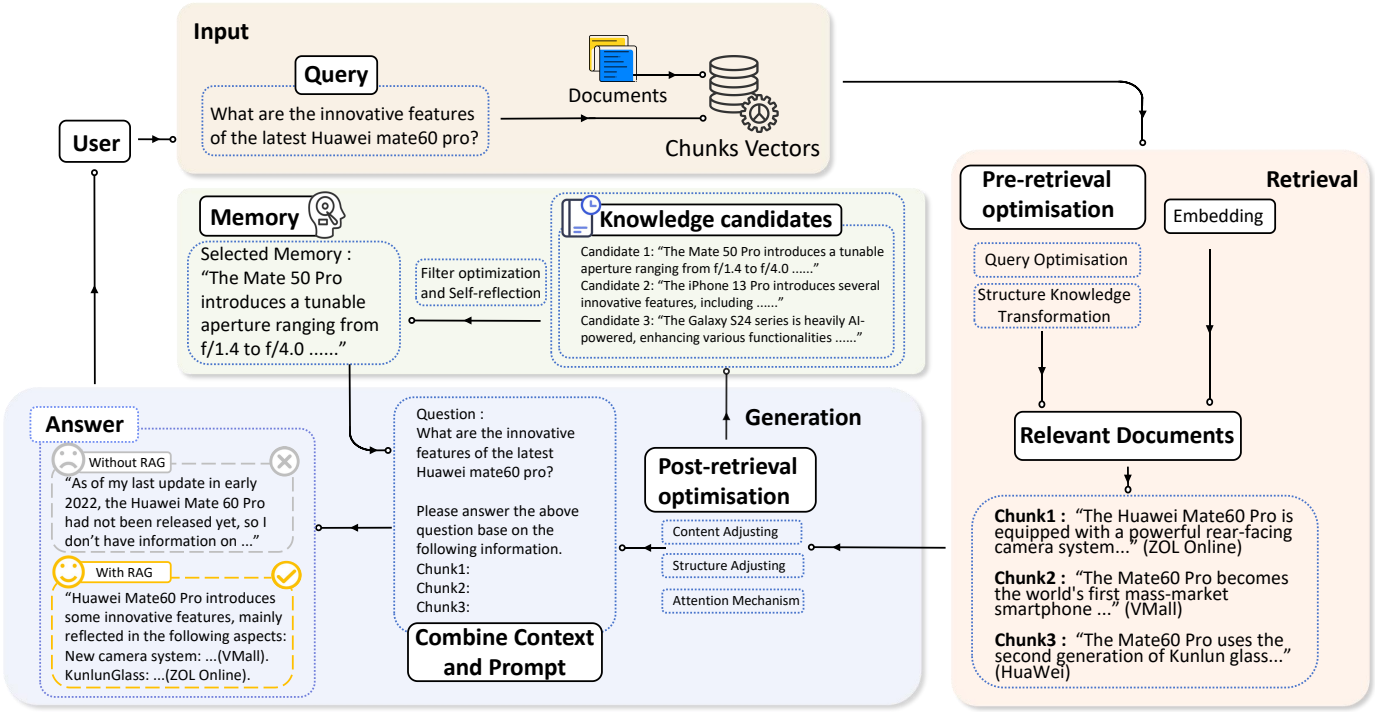


Fig. 6: A workflow of the advanced RAG techniques.

the original query to form a revised version. This process effectively integrates corpus semantics into the user query. Query2doc has demonstrated significant improvements in BM25 performance, boosting it by 3 to 15 percent on specific information retrieval datasets such as MSMARCO and TRECDL, without requiring model fine-tuning. Similarly, ITER-RETGEN has shown enhanced performance across various question-answering tasks, including Natural Questions, TriviaQA, 2WikiMultiHopQA, and HotpotQA, surpassing previous baseline results.

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

a) Content Adjusting. RALM with CoN [60] excels in specifying the reading order for the content retrieved from the search knowledge base. Before finalizing an answer, potential responses are generated from the retrieved content and meticulously evaluated. This evaluation process identifies correlations between the potential responses and the query, selecting the most relevant one. RALM with CoN strategically highlights key sections and modifies the document accordingly, reconstructing its content to better align with the original query. In experimental settings, RALM with CoN's effectiveness was further validated by introducing noise into the retrieved documents, which enhanced the system's noise robustness. Moreover, RALM with CoN demonstrated improved unknown robustness by effectively evaluating queries not covered during pre-training. ARM-RAG [61] takes a different approach, utilizing neural information retrieval to trace reasoning chains, particularly in solving mathematical problems. During their experiments, they found that accuracy could be improved by replacing words that might cause significant shifts in the model's reasoning. This technique involves blurring words that could disrupt the reasoning process, thus improving the model's overall accuracy.

b) Structure Adjusting. Adjusting the structure of retrieved documents can enhance the effectiveness of the model's responses [62]. Articles can be classified into four categories based on their relevance to the query: Gold Documents, Relevant Documents, Related Documents, and Irrelevant Documents, in decreasing order of relevance [62]. Their study demonstrated that incorporating Irrelevant Documents into the document reconstruction process improved performance accuracy by more than 30

c) Attention Mechanism. System2Attention (S2A) [63] enhances the soft attention mechanism in LLMs within the Transformer architecture by refining and refocusing the attention process. Their approach leverages the LLM itself to build stronger attention mechanisms. Specifically, it uses prompts to adjust the LLM, enabling it to reconstruct the retrieved document into a new one by removing irrelevant text. This process simulates human cognitive functions like thinking, reasoning, and decision-making, providing control over attention focus. As a result, the reconstructed document eliminates irrelevant content and highlights essential information. Additionally, S2A introduces further techniques to refine attention. The "2" in S2A refers to generating the final response based on the reconstructed document, essentially refocusing attention one more time. S2A demonstrates promising performance, particularly on the TriviQA dataset, where it outperforms LLaMA 2-70B-chat in factuality with scores of 80.3% compared to 62.8%. On GSM-IC, S2A improves accuracy from 51.7% to 61.3%.

3) Memory. In addition to constructing an effective external knowledge base, RAG can also store retrieved re-

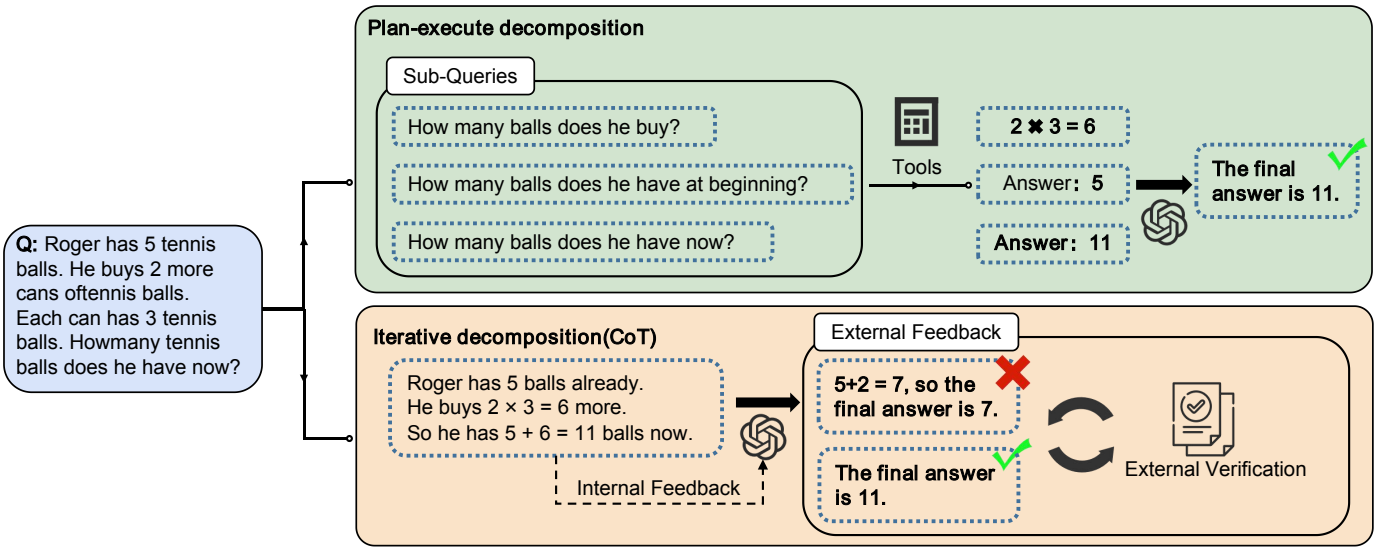


Fig. 7: The target decompesiton of Reasoning.

sults and conversational histories to build memories. These memories are classified into two types: external and internal memories.

a) External Memories. The Selfmem framework [12] employs RAG in an iterative manner to create both a memory pool and a memory selector. This selector identifies an output to serve as memory for subsequent generation rounds. The core concept of this framework is rooted in prompt engineering, where the prompts presented to the LLM are more similar to its own outputs stored in the memory pool than to the original training data. This is due to the fact that, during inference, the memory that most closely aligns with the data distribution is not the training data (which has a BLEU score of 38.89), but rather the model's own outputs within the generative space (which achieves a BLEU score of 58.58). Within the memory pool, multiple rounds of search optimization and retention are conducted to identify and maintain the most representative search results.

b) Internal Memories. Internal Memories leverage the reasoning capabilities of LLM to evaluate and provide feedback on retrieved information, forming the internal memory of the LLM [13]. To implement this, the MetaRAG framework is introduced, which incorporates three core processes: monitoring, evaluation, and planning during inference. MetaRAG has demonstrated significant performance improvements, achieving a 34.6% accuracy increase on the 2WikiMultihopQA dataset and a 26% accuracy improvement on the HotpotQA dataset.

3.3 Reasoning and Planning

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

3.3.1 Target Decomposition Reasoning

Target decomposition is the key prompt technology that enhances the reasoning ability of LLMs. It mirrors the core of human reasoning, where individuals, through experience, learn to tackle complex goals by breaking them down into more manageable sub-goals [64]. This section introduces the essential prompt strategies: Plan-Execute Decomposition and Iterative Decomposition.

A. Plan-execute Decomposition. The approach involves *decomposing* a complex query Q into a series of simpler sub-problems that can be addressed sequentially. This method emphasizes the relationship between problem-solving steps, enabling the inheritance and promotion of previous solutions. By breaking down the query, the underlying focus of the prompt becomes clearer, revealing the essence of the challenging problem. The Least-To-Most Prompting method [65] consists of two main phases: Decomposition and Sub-problem Solving. In the Decomposition phase, the prompt includes examples and specific instructions to illustrate the breakdown process. Subsequently, in the sub-problem Solving phase, attention shifts to demonstrating how each sub-problem is resolved using the provided examples, which are recorded in a list. This list stores previously answered sub-questions along with their corresponding answers, and it helps identify the next question to address in the sequence.

B. Iterative Decomposition. Chain of Thought (CoT) is the core technique of Iterative decomposition. CoT decomposes task goals during reasoning process. It iteratively decomposes the problem to identify and address sub-goals. This process is then repeated until the target fully completed. Finally, the reasoning results are generated. The origin of the CoT within LLMs can be linked to the pioneering ideas proposed by Jason et al. [17]. Their ideas revolve around generating thought chains to enhance LLMs' ability to perform complex reasoning tasks. When LLMs are

provided with limited samples during inference, the prompt follows a structured triplet format: <input, chain of thought, output>. This structured framework equips LLMs with the capacity to produce similar chains of reasoning, offering valuable insights into their computational capabilities and reasoning processes.

1) User-level CoT Prompt. The work by Jason Wei et al. [17] is regarded as the pioneering study on CoT prompting. By providing the LLM with a sequence of intermediate reasoning steps within the prompt, the model can emulate the human problem-solving process, ultimately arriving at an accurate solution. This process can be further simplified through the use of zero-shot and few-shot prompting techniques, as outlined in Section 3.1.3. Additionally, Kojima et al. [42] demonstrate that the inclusion of the phrase ***"Let's think step by step"*** in the prompt enhances the LLM's ability to perform CoT decomposition. Leveraging few-shot examples, LLMs can autonomously decompose and solve complex problems, leading to reductions in both energy consumption and processing time. The scope of the CoT is extensive, encompassing a wide range of problems encountered in daily human life, including mathematical calculations, common-sense reasoning, and more. Notably, CoT, based on few-shot examples, demands less effort and time compared to directly training an LLM.

2) Technique Enhanced CoT. After the introduction of CoT, many researchers begin to focus on how to improve the efficiency and performance of CoT. These optimization efforts often depend on the support of system-level tools. The advancements in CoT optimization can be categorized into three primary areas: enhancing the fundamental principles and explainability of CoT, refining the CoT methodology, and derivation of CoT.

Auto-CoT automates the process of sampling diverse problems and generating inference chains for constructing examples [66]. This framework contains two main components: Question Clustering and Demonstration Sampling. In the Question Clustering phase, diversity-based clustering is employed to mitigate misdirection by ensuring similarity among the sampled problems. Initially, for a given query group Q, each problem is represented as a vector using Sentence-BERT [67]. These vectors are then clustered using the K-Means method. Following clustering, distinct clusters are formed, each comprising a list of queries, rationales, and corresponding answers. For each cluster, the problem closest to the center of the cluster is selected from each class within the cluster. These selected problems are then incorporated into inputs, which are subsequently processed using the Zero-Shot-CoT approach [42] to generate the inference chain by LLMs.

Active Prompt [68] addresses the challenge of adapting CoT manual annotation examples to diverse tasks. It enables LLMs to adjust to task-specific requirements through example prompts annotated with manually designed CoT inferences. The Active Prompt framework comprises four key stages: Uncertainty Estimation, Selection, Annotation, and Inference. In the Uncertainty Estimation phase, the LLM is probed by generating potential answers to intermediate steps across multiple iterations (k times) on a set of training problems. Computational uncertainty analysis is then applied to the responses. The problem exhibiting the highest uncertainty is selected for further analysis. During the Selection phase, the most uncertain questions are identified and manually annotated in the Annotation phase. Finally, each problem was extrapolated with new annotated examples to further refine the model's understanding.

The Faithful CoT framework [69] is designed to enhance the interpretability of CoT reasoning, thereby improving both its reasoning accuracy and persuasiveness. Faithful CoT comprises two main components. First, it involves decomposing the original query (Q) into an inference chain (C) by using prompts to guide the model. Subsequently, a natural language (NL) program breaks down C into several simpler, interrelated subproblems. Second, the solution for each subproblem is determined by a symbolic logic (SL) program, typically implemented in languages such as Python, Datalog, or PDDL. The execution of SL and NL programs occurs alternately, without a strict order. Faithful CoT offers superior interpretability and demonstrates significant empirical performance improvements over standard CoT. It outperforms standard CoT on nine out of ten benchmark tests across four diverse domains. Notably, it achieves a relative accuracy gain of 6.3% for mathematical word problems (MWP), 3.4% for planning, 5.5% for multi-jump question answering (QA), and 21.4% for relational reasoning. Furthermore, when applied with advanced models such as GPT-4 and Codex, Faithful CoT sets new state-of-the-art results for low-sample performance across seven datasets, with six achieving accuracy rates exceeding 95%. These results highlight the strong synergy between interpretability and accuracy that Faithful CoT brings to CoT reasoning tasks.

In addition to research on CoT itself, many scholars have begun exploring alternative methods for constructing reasoning chains. Building upon CoT, a more comprehensive and generalized framework known as XoT [70] was proposed. This framework provides a broader variety of reasoning chains compared to CoT. XoT allows for switching between different modes of reasoning chains and can adaptively switch between them when the model encounters obstacles. The switching mechanism between reasoning chains in XoT is governed by two validation methods. Passive validation performs basic checks, such as ensuring there are no compilation errors, while active validation requires the model to assess whether the answers meet the conditions specified in the query (Q). XoT can also take on various structural forms. The Tree-of-Thought [18] approach organizes reasoning chains into a tree-like structure. The Graph-of-Thought [19] transforms reasoning chains into a graph structure, further expanding the flexibility and applicability of the reasoning process.

3.3.2 Tools and Feedback Enhanced Reasoning

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

A. External Support. LLMs are expected to perform both semantic understanding tasks (e.g., task intent understanding) and complex reasoning tasks (e.g., numerical

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

1) Experimental Simulator. Current LLMs face challenges when dealing with realworld problems that require a deep understanding of physical principles [71]. Due to the limitations of LLMs, LLMs' responses are often based on the semantic interpretation of the text rather than the rules of physics, making it difficult for them to reason logically through existing knowledge [72, 73]. As a result, relying solely on LLMs for solving physical problems is problematic. To address this limitation, it is crucial to conduct simulation experiments on physical problems and incorporate the results into LLM-generated responses.

Mind's Eye [71] leverages a computational physics engine [74] to simulate real-world physical processes. a Text-to-Code language model is employed to generate rendering code for the physics engine, allowing the simulation of physical experiments relevant to the posed question. The simulation outcomes are then incorporated into the LLM prompt in natural language, compensating for the model's lack of physical understanding. Mind's Eye demonstrated a significant improvement in inference accuracy, achieving increases of 18.4% and 34.2% in zero-shot and few-shot settings, respectively, compared to relying solely on the language model.

2) Code Interpreter. When confronted with complex reasoning problems, offloading the precise computation task to external modules can significantly improve the model's solution accuracy. Drori et al. [75], Chen et al. [76] prompt Codex [77] to generate executable code-based solutions for problems ranging from university-level exercises to mathematical word problems and financial question-answering (QA). This idea of cooperation between LLMs and code has also been applied to solve more specific problems [78, 79]. One possible interpretation of these findings is that code-based approaches benefit from well-defined structural consistency, offering advantages in robustness and logical reasoning compared to natural language. This form of code-driven reasoning allows LLMs to focus more on problem-solving logic rather than the intricacies of textual representation.

The PAL method [80] employs a CoT prompting strategy to decompose complex symbolic reasoning, mathematical problem-solving, and algorithmic tasks into intermediate steps, represented through Python code and natural language annotations. In this setup, computational tasks are delegated to the Python interpreter. Similarly, Chen et al. [81] introduce "Program of Thoughts" (PoT) prompting, which separates computational and reasoning processes through specific prompt designs. Unlike PAL, PoT follows a zero-shot approach for prompt generation. Luo et al. [82] propose a framework named MultiPoT to select the optimal external programming language based on task types to overcome the Python language's extension limitations. MultiPoT creates custom hints for each programming language to ensure semantic consistency and structural diversity. When addressing issues, it integrates multiple programming languages and selects the final answer generated from each PL by self-consistency. The results obtained incorporate the benefits and diversity of multiple programming languages.

3) Integration Tools. In addition to code interpreters and simulators, more tools have been collected in the form of APIs. Integration tools are the frameworks that incorporate diverse tools into prompts. The strong generalization ability of LLM allows it to effectively utilize natural language as an intermediary to manipulate external tools [83]. Additionally, prompt technologies serve as the guides for these tools, providing instructions and guidance on how to effectively utilize them. Based on this insight, Parisi et al. [84] build a text-to-text_api prompt enhancement framework (TALM). The TALM bootstrap LLM generates '—tool-call' and 'tool input text' to construct API call requests, and then appends the results returned by external api's to the text sequence, the model's capabilities are significantly expanded. Similar to this, Toolformer [85] uses the same approach of including API requests in the prompts to leave the sub-operations to external tools, while Toolformer also suggests ways to build datasets and fine-tune them. Galactica [86] designed a special token to initiate a request to call an external tool. This work proposed the concept of 'work memory', which generates a special segment in which algorithms and problem-solving code are generated when the model needs to call an external tool. These methods require a strategy to determine what tasks should be offloaded.

To overcome the limitations of manually writing API requests, some works propose strategies to make LLM use tools automatically. MRKL [87] proposes a modular neural-symbolic architecture that divides tasks into corresponding task APIs through a router and integrates the returned results. ART [88] organizes its own task library, retrieves tasks related to the original prompt in the task library, then decomposes a series of tasks in turn into corresponding sub-tool sequences, generates a specific cue word to be processed, and finally integrates the results of the tool API into the original cue word to realize the automated tool usage of LLM. More large-scale architectures such as Chameleon [89], Gorilla [90], HuggingGPT [91], ToolAlpaca [92] These large-scale architectures take advantage of richer API tools and more systematic API requests, greatly enhancing the ability of LLM to use tools to handle tasks. Based on these architectures, a number of LLM + Tool agents such as TPTU [93], TPTUv2 [94], TaskMatrix.AI [95] have also demonstrated a strong ability to handle complex tasks in different scenarios.

B. Feedback. As is the case with human learning, feedback can assist individuals in recognising areas for improvement and addressing issues more effectively. The provision of feedback enables the LLM to ascertain the correct and incorrect responses through prompting, thereby enhancing its capacity for reasoning. In accordance with the subject of the judgment, feedback can be classified into two distinct categories: internal feedback and external feedback.

1) Internal Feedback. LLMs possess the inherent capability to engage in self-feedback. Numerous scholars in this

field have demonstrated the self-feedback ability of LLMs and have proposed corresponding methods to enhance and leverage this mechanism effectively. By harnessing self-feedback, LLMs can iteratively refine their own outputs, contributing to their continual improvement and better performance.

a) Internal Feedback Techniques. Some scholars [96] demonstrated that LLMs can undergo iterative self-refinement without requiring additional training, introducing the SELF-REFINE approach. This method refines the output of LLMs through alternating iterations of feedback and refinement to enhance the output quality. The feedback generation process is guided by a few-shot prompt [3]. SELF-REFINE outperforms models such as GPT-3.5 and GPT-4, directly yielding absolute improvements ranging from 5% to 40%. In tasks involving code generation, when applied to CODEX, SELF-REFINE improved initial generation by up to 13%.

Self (Self-Evolution with Language Feedback) [97] enables LLMs to engage in self-feedback and self-refinement. This is achieved by providing LLMs with feedback in natural language, empowering them to autonomously evolve. The evolution process involves generating responses to unlabeled instructions and iteratively refining these responses through interactions. SELF teaches LLMs fundamental meta-skills using a limited set of examples in natural language, fostering a continuous cycle of self-evolution for LLMs. Self-Contrast [98] enhances the self-feedback capacity of LLMs. This approach prompts LLMs to generate various perspectives to address the same problem, subsequently facilitating a comparison and reflection on the disparities among these perspectives.

Some scholars [99] proposed that LLMs can autonomously provide self-feedback regarding hyperparameter perception. Through hyperparameter aware instruction tuning, LLMs ascertain the optimal decoding strategy and configuration based on input samples, thereby achieving self-regulation. Their proposed HAG (Hyperparameter Aware Generation) consists of two components. Firstly, by inputting a query Q to the LLM, HAG generates appropriate hyperparameters. Subsequently, HAG instructs the LLM to adjust the model's decoding strategies and hyperparameters according to the generated parameters, culminating in the generation of the final result post-adjustment.

b) Evaluation of Internal Feedback. MINT benchmark [100] is designed to facilitate the acceptance of natural language feedback by LLMs, simulated by GPT-4 users. This benchmark serves as an effective means to bridge the connection between LLMs and LLM-tool interactions. Moreover, the benchmark is capable of accessing Python code execution tools, enhancing its utility and versatility.

Some scholars [101] proposed that glass-box features should be taken into account in self-assessment of LLMs. They propose that the softmax distribution serves as a dependable indicator for quality evaluation. Their research reveals a strong correlation between manipulating the softmax distribution—by calculating the entropy and variance of the softmax—and annotated evaluation results.

Self-feedback can enhance performance on certain tasks for LLMs, while potentially worsening performance on others [102]. In light of this, some scholars [103] introduced the concept of Self-Bias to evaluate the bias of LLM output accuracy across different tasks. Additionally, their research reveals that models with larger parameters and access to external feedback possess the ability to more accurately assess and mitigate Self-Bias.

2) External Feedback. Certain external tools can verify the extrapolation outputs of large language models (LLMs). CRITIC framework [104] enables appropriate tools to assess the output of LLMs, which in turn allows LLMs to adjust and refine their output based on this feedback. These external evaluation tools encompass search engines, code interpreters, text APIs, and more. The feedback process primarily involves validating the initial output expectations generated by the LLMs and then modifying the output based on critiques from the verification process. Through this iterative process, external tools offer feedback to LLMs from an external perspective, thereby enhancing the performance of LLMs.

3.4 Reliability

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

In the first three sections, we introduced how to design an effective prompt for LLMs using the *Profile-Knowledge-Reasoning* process. Due to the uncertainty of the LLM's response [105], the same prompt can lead to diversified responses from the LLM. At the same time, due to the LLM's context learning ability, it is particularly sensitive to the order, type, and historical bias of the prompts [106, 107, 105]. The bias of the prompt may worsen the performance of the LLM in downstream tasks. Furthermore, the reasoning strategy used by the LLM is opaque, which means that the responses of the LLM are not always trustworthy [108], and there are a series of risks associated with the responses of the LLM [109, 110]. In practical applications, users often demand stable and reliable responses from the LLM. The process of reducing the bias of the LLM's response through prompt integration or by generating auxiliary knowledge from the LLM is known as improving the reliability of the LLM. In this section, we will explain from both content bias and value bias perspectives.

3.4.1 Content Bias

We define content bias as the deviation between the completion results of the LLM and the task requirements. For most LLM tasks, designing prompts according to the Profile-Knowledge-Reasoning process can help the LLM perform well in response to task requirements. However, when the task is more complex, it is difficult to obtain stable and perfect output through a single round of prompt input [107, 109]. An intuitive idea is to adjust the order of the prompts or change the method of prompting to generate multiple rounds for the same question, which actually uses the concept of ensemble. The prompt method based on this idea is called prompt ensembling.

A. Prompt Ensembling. Prompt ensembling refers to the use of multiple different prompts to complete the same task, enhancing the reliability of the results through multiple responses. Prompt ensembling borrows from the pattern of ensemble learning and includes two sub-processes: first generating multiple prompts as input, and then combining the responses of multiple prompts through a specific strategy to obtain the final result [111, 112]. Bagging and boosting are two typical ensemble methods widely used in many classic tasks and have unique applications in LLM.

1) Bagging Prompt. The application of Bagging Prompt in LLM mainly falls into two categories: the majority vote method based on Self-consistency and the beam search method based on Step-Verifier [113]. BPE [114] focuses on constructing few-shot CoT prompts based on self-consistency [115], which outperforms a single prompt. However, since Self-consistency is a method based on a greedy approach, it cannot guarantee that the inference chain is entirely correct. Also, its voting is atomic, which means it cannot distinguish the quality of different responses.

In response to these challenges, DiVeRSe [116] was developed to enhance answer reliability through a three-stage process: 'generate-verify-check.' First, it generates diverse completions using multiple prompts. Next, a 'step-aware voting verifier' model distinguishes good answers from poor ones and verifies the correctness of inference steps. DiVeRSe extends traditional methods by extracting step-level labels from intermediate results, ensuring accuracy in the inference flow. While it integrates the benefits of Bagging Prompt methods, DiVeRSe still relies on manual sample selection, and sample bias can affect final results.

Motivated by these issues, the AMA (Ask Me Anything) method was introduced [117]. AMA consists of two stages: the multiple prompt step and the answer aggregation step. In the multiple prompt step, AMA employs a functional prompt chain where the *question()* function transforms the input into open-ended questions, providing diverse perspectives for LLMs to address different aspects of the problem. The *answer()* function then generates intermediate answers. AMA highlights the limitations of simple voting due to equal weighting and question similarity, which can distort results. To overcome this, AMA uses an answer aggregation method based on weakly supervised learning with an information entropy penalty, ensuring that results better reflect different perspectives. Essentially, AMA optimizes the Bagging random sampling approach.

2) Boosting prompt. Boosting prompt methods often adopt a two-stage paradigm. PromptBoosting [118] applies the AdaBoost algorithm on the prompt set, achieving good results in text classification. However, the PromptBoosting method requires a pre-prepared high-quality prompt set and cannot optimize for specific prompts.

To overcome these limitations, Prefer [119] established a feedback mechanism to reflect on the shortcomings of the weak learners in the current iteration. Based on feedback, Prefer also implements the automatic synthesis and selection of prompts, avoiding the bias problem brought by the prompts. PromptBoosting's ensemble method for weak learners refers to the traditional ensemble method of weighted summation. However, many works have pointed out that LLMs have a serious optimistic estimation problem [120, 121]. This makes the weighted calculation of answer accuracy unable to eliminate content bias.

Bilateral Prompt Bagging [119] assesses the confidence of the generated results in each iteration. When the assessment result of the answer is not trustworthy, a reverse confidence evaluation is performed, calculating the confidence that this answer is incorrect. The final correct probability of a round of generated results is evaluated by combining forward and reverse confidence. This design of positive and negative confidence effectively avoids the optimistic estimation problem of large models [120], enhances the effect of single-round generated results by utilizing the feedback reasoning ability of large models (introduced in Section 2.3), thus improving the ability of weak learners and making the final result more reliable and efficient.

3.4.2 Value Bias

We define value bias as the deviation between the content generated by LLMs and human societal values. The content generated by LLMs often touches upon socially sensitive areas, such as issues of social harm and discrimination [122, 123]. Shaikh et al. [109] show that Chain of Thought (CoT) can continually enhance the performance of LLMs across a variety of NLP tasks. However, this also increases the likelihood of the model producing harmful or inappropriate results. Research indicates that CoT prompts should be used carefully when dealing with socially relevant issues. Due to the principles of autoregressive decoding inherent in LLMs, harmful text might still be generated during the beam-search phase of the inference stage if the volume of harmful text in the training corpus is large enough. This underlines the importance of developing mechanisms to manage and minimize harmful or socially unacceptable outputs for the safe deployment of AI systems.

Tang et al. [124] propose the Detox-Chain method which links detoxification sub-steps together to achieve rapid detoxification of prompts. Initially, the Toxic Span Detection step is used to identify the toxic regions in the input prompt. Subsequently, the Toxic Span Repair step masks and iteratively processes these toxic regions, transforming the toxic textual expression into a safe one. Finally, the Text Continual Generation stage completes the prompt by filling in the replaced parts continuously. The Detox-Chain method reduces the possibility of LLM generating toxic texts by substituting toxic textual representations. However, due to the detoxification technology for prompts altering the distribution or content of the prompt to varying extents, it affects the quality of the output produced by the large model. At the current stage, the detoxification methods for large models are still primarily based on reinforcement learning or knowledge editing techniques during the training phase.

4 Applications

In this section, we aim to provide practitioners with an overview of the various application areas of LLM based on prompt engineering. The above (Section 3) introduces a new taxonomy of LLM prompt engineering based on the logic of human problem-solving. In this section, we propose a new taxonomy of LLM applications based on the logic of prompt

engineering technical classification, including LLM capabilities and scenarios in which users use LLM. We categorize the applications of LLMs into two categories: Cognitive Applications of LLMs and Transformative Applications of LLMs. This taxonomy aligns with the classification framework of prompt engineering, as illustrated in Figure 1. It facilitates a seamless mapping between prompt engineering techniques and their practical applications.

4.1 Cognitive Applications of LLMs

The cognitive application of LLM refers to the fact that LLM provides its own knowledge to the outside world and uses its large number of parameters to analyze the information in depth. LLMs can serve as the object of knowledge, providing users with information, a process we call **information acquisition**. Applications such as *chatbots*, *professional knowledge Q&A*, *search engine*, and *training data augmentation* are all uses of the large language model's application to provide knowledge. Similarly, LLM can analyze the information in the interaction, which can reveal potential information. Applications of LLM analyzing information include simulating *financial markets*, and exploring *chemical molecules*.

4.1.1 Information Acquisition

A. Chatbots. General chatbots (dialogue agents) combine tasks such as information retrieval, multi-turn interaction, and text generation (including code). Large language models store a vast amount of knowledge in their parameters during the training phase. The application of users obtaining information from LLMs through role-playing dialogue is a chatbot[125].

Glaese et al. [126] proposed Sparrow, a dialogue agent based on a 70B parameter Chinchilla LLM. Sparrow uses RLHF (Reinforcement Learning with Human Feedback) to fine-tune the model according to 23 rules aligned with human societal norms, making it more helpful, accurate, and harmless. Sparrow uses prompt engineering techniques that include Basic RAG Techniques (Section 3.2.1) and Personality Information (Section 3.1.1). To address the hallucination issues of large models, Sparrow utilizes a basic RAG model combined with external knowledge, providing evidence from Google search queries. Sparrow also supplements personal information by setting system prompt to provide helpful information when talking to users.

Similarly, OpenAI[127] utilized supervised fine-tuning with high-quality proprietary data and high-quality question-answer corpora, along with RLHF (Reinforcement Learning from Human Feedback), to specialize the GPT-3.5 LLM. The trained model served as the foundation for the ChatGPT chatbot. GPT-4 [128] is the underlying model for the ChatGPT Plus chatbot. ChatGPT uses the prompt technology of Personality Information (Section 3.1.1) to implement role-playing conversations and Memory (Section 3.2.2) technology to record details and highlights of conversation history, thus performing well in long discussions. Microsoft Copilot [129] is an AI-powered productivity tool that coordinates LLM with Microsoft 365. Microsoft Copilot is based on OpenAI model support, uses more integration tools (Section 3.3.2), can browse the web through Bing to get the latest information, and combines RAG technology (Section 3.2) to enrich the user's original prompt.

[130] introduced the Claude series of chatbots, which acquired foundational linguistic capabilities through pre-training on a vast and diverse dataset. The bots were further refined via fine-tuning with high-quality data and guided by RLHF to generate responses that are beneficial, harmless, and honest. Claude also used many of the prompt techniques mentioned above to improve performance. The Claude 3.5 Sonnet model has outperformed other competitor models in most common AI system evaluation benchmarks. These tests include undergraduate-level expert knowledge (MMLU), graduate-level expert reasoning (GPQA), basic mathematics (GSM8K), etc. Opus demonstrates near-human understanding and fluency, leading other models in performance on complex tasks.

B. Search Engine. Information Retrieval (IR) systems are a major component of dialogues, question answering, and recommendation systems, and are our primary means of obtaining information. The integration of LLM with IR systems effectively helps IR systems capture more detailed information.LLMs can be used to enhance traditional IR components, such as query rewriter, retriever, reranker, and reader. LLMs can also serves as the engine of Generative Retriever.

The query rewriter adds synonyms or related terms to the original query in order to address vocabulary mismatch issues and clarify ambiguous queries, thereby more accurately aligning them with the user's intent. In conversational retrieval, the query rewriter understands the context of the entire conversation, clarifies ambiguous content, and generates a more effective new query based on the user's dialogue history. The LLMs can enhance the IR system using a variety of prompt engineering techniques. HyDE [131] represents pioneering work on LLM-based query rewrite. HyDE guides LLM generation using a variety of prompt engineering techniques in Profile and Perception (Section 3.1). HyDE generates more detailed hypothetical documents based on the given query through LLM. These documents are then retrieved from the corpus using a dense retriever. Query2doc [132] represents another pioneering work on query rewriting, which generates pseudo-documents by prompting LLMs with a few demonstrations, which reflects the use of the Demonstration Information approach (Section 3.1.2). These pseudo-documents are then expanded with the generated documents. Based on the optimization idea of prompt engineering, Jagerman et al. [133] studied the impact of different prompting methods and different model sizes on query rewriting [133]. GFF [134] proposed a "generate, filter, and fuse" method for query expansion, which uses LLM to draw a group of related keywords from the original query through a reasoning chain. GFF uses the strategy of prompt ensembling (Section 3.4.1) to filter the generated keywords through techniques such as Self-consistency to ensure the quality and relevance of keywords. The keywords are integrated with the original query for downstream reranking tasks.

The generative retrieval method uses a unified model to directly generate document identifiers (i.e., DocIDs) related to the query, and these methods are also combined with prompt engineering. Traditional information retrieval

systems usually follow the "index-retrieval-reranking" paradigm to locate documents related to a user's query, which has been proven effective in practice [131, 135]. However, the constituent modules of these systems: the indexing module [136, 137], the retrieval module, and the re-ranking module [138, 139] operate independently. LLM-based generative retrieval unifies the modules and initiates a new paradigm of IR systems. Researchers have found that LLMs (such as the GPT series of models) can directly generate URLs related to user queries [140], due to the large amount of HTML resources that the LLM comes into contact with during the training phase. This inherent ability allows the LLM to act as a generative retriever, directly generating document identifiers from the original input to obtain relevant documents. Ziems et al. [140] introduced the LLM-URL model. It uses the GPT-3 text-davinci-003 model to generate candidate URLs. It uses the Demonstration Information approach (Section 3.1.2) to standardize the model output. Additionally, it has designed a URL filtering mechanism that extracts valid URLs from these candidate URLs using regular expressions, thereby locating the retrieved documents.

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

LawGPT [141] is a robust language model designed to address legal knowledge inquiries with high accuracy. It employs the LLM internal feedback method (Section 3.3.2) to continually enhance its precision in legal question answering. At the same time, the personality information (Section 3.1.1) method is used to determine the legal knowledge location for LLM. Through this approach, LawGPT generates legal queries pertinent to specific legal texts and subsequently provides responses in the form of "text segment-question" pairs, thereby ensuring that its answers are rich in legal information.

MultiMedQA [142] is a benchmark that integrates clinical expertise with medical knowledge, employing a Few-shot prompting technique (Section 3.1.3). Collaborating closely with a panel of seasoned clinicians, MultiMedQA produces exemplary few-shot demonstrations and sample scenarios. Moreover, it adeptly demonstrates Reasoning and Planning capabilities, exhibiting a robust chain-of-thought process (Section 3.3.1) for a myriad of medical issues by leveraging insights from various medical professionals. Furthermore, MultiMedQA excels in internal feedback (Section 3.3.2) verification, a crucial aspect in refining its accuracy. Given the multifaceted nature of medical queries, it employs the self-consistency strategy (Section 3.4.1), allowing the model to compare and evaluate diverse perspectives to ensure coherence and reliability in its responses.

D. Training Data Augmentation. Due to the high cost of manually annotating labels, a common problem in training neural retrieval models is the lack of training data. LLM can learn the patterns of data from manually annotated data and generate additional data that is consistent with the existing dataset.

Yoo et al. [143] proposed GPT3Mix, which generates synthetic data from existing datasets based on GPT3 LLM. GPT3Mix uses the Demonstration information method (Section 3.1.3) and the Task Inforamtion method (Section 3.1.2). The prompts of GPT3Mix include two parts: real examples from the dataset and task specifications, which are used to create synthetic data and pseudo labels. Yoo et al. [143] used this new augmented dataset to fine-tune BERT and DistilBERT models, achieving excellent performance in classification tasks. In the context of information retrieval, it is easy to collect many documents. However, the challenging and expensive task is to collect real user queries and accordingly label the relevant documents. Given the LLM's powerful natural language processing capabilities, many researchers [144, 145] suggest using an LLM-driven process to create pseudo queries or relevance labels based on existing datasets. In cutting-edge work, Dai et al. [146] introduced ChatAug. ChatAug rephrases training samples from a small dataset into multiple conceptually similar but semantically different samples using six rewriting examples (Demonstration information method), enhancing the consistency and robustness of the data. ChatAug performs better than the state-of-the-art text data augmentation methods in terms of test accuracy and augmented sample distribution.

4.1.2 In-depth Information Analysis

A. Financial Markets. By providing internet-scale data for LLMs, LLMs can use hybrid training methods in financial tasks, driving the open-source development in the financial field.

BloombergGPT [147] is a formidable large language model boasting 5 billion parameters, meticulously trained to excel in the intricate realm of finance through vast datasets. BloombergGPT combines prompt engineering technologies such as personality information (Section 3.1.1) and external financial tools (Section 3.3.2) to provide more personalized, time-sensitive financial advice. Its versatility extends across various tasks within the financial sector, making it beneficial for professionals. Leveraging techniques such as few-shot learning and other sophisticated methodologies, scholars fine-tune BloombergGPT to suit specific task formats, thereby enhancing its efficacy in delivering best-in-class results. FinGPT [148] stands out as an open-source, large language model tailored specifically for the financial sector, boasting versatile applications such as robot-consultation, algorithmic trading, and low-code accessibility. Its remarkable capability lies in making quantitative inferences from vast financial datasets, effectively capturing the intricate dynamics of the financial markets. A key feature of FinGPT is its adept utilization of external feedback methods (Section 3.3.2) within prompt engineering. By leveraging stock prices as indicators, FinGPT engages in reinforcement learning, continuously refining its understanding and interpretation of financial texts. This process empowers FinGPT to anticipate and predict market responses to a wide array of financial events, thereby enhancing its overall performance and utility in financial analysis and decision-making.

B. Chemical Molecules. Chemical molecules study the properties, composition, structure, and chemical reactions of matter. The field involves many complex computational and predictive tasks, such as property prediction, chemical structure optimization, and reaction prediction. As data

scales increase, traditional chemical computational methods can no longer meet the demand. Therefore, the application of LLMs in the field of chemistry is becoming increasingly important.

BioinspiredLLM [149] is an autoregressive transformation large language model specifically tailored for the realm of biomaterials and biomimetic materials science. This sophisticated model excels in accurately recalling vast amounts of information pertaining to biomaterials, effectively reflecting intricate patterns between various biomaterials, and facilitating research tasks within this field. Scientific researchers continuously uncover new patterns, and BioinspiredLLM accommodates this by offering additional contextual content when queried. Moreover, the model leverages the Retrieve and Generate (RAG) framework, based on Knowledge prompt engineering techniques (Section 3.2), to provide more comprehensive and insightful answers.

ChatDrug [150] harnesses conversational and reasoning capabilities to facilitate AI-assisted drug discovery endeavors. ChatDrug uses CoT technology (Section 3.3.1) to decompose complex concepts into more understandable attributes, thereby improving reasoning ability and problem-solving efficiency. Moreover, ChatDrug incorporates prompt design tailored for domain-specific modules, with the flexibility to implement prompt functions via few-shot learning methodologies (Section 3.1.3). Leveraging the Retrieve and Generate (RAG) method (Section 3.2) further enriches ChatDrug's retrieval capabilities by augmenting its dataset, thereby enhancing the quality of responses. Additionally, ChatDrug demonstrates a robust internal feedback mechanism (Section 3.3.2), which bolsters its accuracy through domain feedback functions. In instances where the output requires validation, ChatDrug iteratively generates new drugs, ensuring continual refinement and improvement in its performance.

ChemCrow [151] enables reasoning across common chemistry tasks such as materials design and synthesis, from reasoning about simple drug discovery cycles to planning the synthesis of substances across a wide range of molecular complexity. ChemCrow can combine the inferential power of LLM with the chemistry expertise of computational tools (Section 3.2) and has been able to plan and synthesize an insect repellent, three organocatalysts, and guided the screening and synthesis of a novel chromophore with target properties. Enhanced by RAG method (Section 3.2) and search engine tools (Section 3.3.2), ChemCrow can obtain scientific information in the Internet and build relevant databases. ChemCrow can also accept external feedback from humans and adjust itself through feedback.

4.2 Transformative Applications of LLMs

LLMs affect society in various application scenarios, which we divide into those that affect the physical world and those that affect the spiritual world. Through prompt engineering, LLM can impact the physical world through a variety of application scenarios. These areas include software engineering, robotics, and expert-level task automation. In these areas, LLMs have contributed to practical progress by enhancing automation, optimizing processes, and improving efficiency. On the other hand, the impact of LLM on the cognitive world covers the realm of human art and creation. This includes applications in language generation, visual content creation, and auditory processing. Using LLMS, advances can be made in areas such as natural language processing, image generation, and audio analysis, thus helping to enrich human creativity and comprehension.

4.2.1 Physical World Applications

A. Software Engineering. One of the most advanced and widely used applications of LLM is to generate and complete computer programs in various programming languages. This section will discuss programming-specific LLM, that is, LLM that are fine-tuned or pre-trained specifically for programming applications. However, it should be noted that general chatbots, which are partially trained on code datasets (such as ChatGPT), are increasingly being used in programming tasks.

code generation is one of the most important and widely used applications in software engineering. Code generation refers to the use of LLM to output new code based on the specifications or questions provided in the prompts. Currently, several LLMs and methods have been proposed for computer programming. OpenAI first released CodeX [77]. This is an LLM based on GPT-3 (up to 12B parameters) and pre-trained on public datasets to generate independent Python functions from NLP strings. CodeX standardizes the output using prompt techniques such as few-shot prompting (Section 3.1.3) and personality information (Section 3.1.1). On the HumanEval evaluation set, the performance of the CodeX model was superior to similarly sized GPT-3 and GPT-J models, with the Codex model trained on the filtered dataset (CodeX-S) achieving the best results. Importantly, Chen et al. [77] pointed out that using a pre-trained GPT-3 model as a base yielded no observed improvements other than faster convergence. The Copilot [129] plugin based on CodeX has also become a benchmark for code generation assistance tools. The HumanEval dataset proposed in the CodeX paper has also become one of the commonly used benchmark datasets for subsequent code generation.

Nijkamp et al. [152] sequentially trained CodeGen series LLMs (up to 16B parameters) on three datasets: the natural language dataset (THEPILE), the multilingual programming source code dataset (BIGQUERY), and the single-language Python dataset (BIGPYTHON). CodeGen models have been trained in a variety of programming languages, covering C, C++, Go, Java, JavaScript, and Python. The results show that the largest CodeGen model trained on a single-language dataset outperforms the Codex-12B model. CodeGen can use CoT technology (Section 3.3.1) to improve the reasoning ability of generated code. Nijkamp et al. [152] also tested CodeGen for multi-step program synthesis, decomposing the program into multi-step natural language prompts, with the synthesis system completing the synthesis of subroutines at each step. At the same time, they created a multi-round programming benchmark (MTPB) for the multi-round program synthesis method. Due to the excellent performance in multiple programming languages, Zheng et al. [153] trained CodeGeex on three datasets: The Pile, CodeParrot, and the public repository supplement data on GitHub. CodeGeex uses a standard Transformer architecture, which can support high-precision code generation,

and at the same time supports automatic translation and conversion of code snippets between different programming languages.

B. Robots. Multimodal large language models are finding application in robotics, where robots interact with the physical world through various physical methods. These models leverage their reasoning and planning capabilities to guide the actions of the robot.

PaLM-E [154], a general visual language model, incorporates embedded data into multimodal training, serving as an effective inference engine. Experimental results demonstrate that aside from general visual language tasks, PaLM-E excels in tasks such as entity capture and overlaying, performing admirably in real desktop and mobile manipulation scenarios. Fine-tuning of PaLM-E involves scaling the language model size, enhancing its adaptability across different tasks and environments. This multimodal approach ingeniously employs prompt engineering across diverse tasks and scenarios. Task information (Section 3.1.2) enables PaLM-E to discern the attributes of various tasks. Task environment information provides task scenarios tailored for specific tasks. Additionally, PaLM-E utilizes Target decomposition reasoning (Section 3.3.1) to break down input tasks in advance. This step-by-step strategy enables the multimodal model to provide more realistic and effective responses.

C. Expert-Level Task Automation. As a Large Language Model with specialized functions tailored to diverse legal tasks such as legal consultation, crime prediction, and court opinions. WisdomInterrogatory [155] is adept at responding to new legal issues using existing legal documents and provisions within the field of law. Through task information (Section 3.1.2), it can use the corresponding legal knowledge to answer the consultation. WisdomInterrogatory systematically clarifies attribute information, information sources, task types, and answer plans for various legal scenarios. The WisdomInterrogatory utilizes RAG technology (Section 3.2.1) for retrieving relevant legal information and enhances the relevance through pre-retrieval optimization (Section 3.2.2). By prompting the LLM with few-shots (Section 3.1.3), WisdomInterrogatory can generate different responses tailored to specific legal tasks. This approach ensures that WisdomInterrogatory can effectively adapt to a wide range of legal scenarios, providing accurate and insightful answers while leveraging its comprehensive understanding of legal documents and provisions.

ChatLaw [156] is a comprehensive legal language model crafted from a deep understanding of the legal domain. It stands as a robust tool for navigating complex legal issues, offering nuanced insights and practical guidance derived from its comprehensive knowledge base and advanced computational techniques. It addresses real-world legal challenges by synthesizing legal awareness, relationships, behaviors, and other phenomena within the legal realm. Leveraging Prompt engineering within the legal domain, ChatLaw significantly enhances the performance of the Chinese legal language model. During the construction of its training dataset, ChatLaw meticulously fine-tunes and supplements established data sets to ensure relevance and accuracy. ChatLaw employs RAG technology (Section 3.2) to retrieve the latest legal information. Additionally, it utilizes post-retrieval optimization (Section 3.2.2) to further refine its responses and improve overall effectiveness.

ChatDoctor [157] is a sophisticated language model designed to address medical queries and offer advice in medical contexts. Developed using a dataset of 100,000 patient-physician conversations sourced from a widely-used online medical consultation platform, this model has been finely-tuned to comprehend patient concerns and provide informed recommendations to aid patients in seeking appropriate medical care. One of ChatDoctor's notable features is its external knowledge brain, akin to a Retrieval Augmented Generator (Section 3.2). This knowledge repository encompasses a vast array of information on diseases, symptoms, relevant medical tests, and more. Continuously updated and refined, the knowledge brain constantly retrieves new insights from sources such as encyclopedias, medical literature, and other credible references. With its robust dataset and access to a rich knowledge base, ChatDoctor serves as a valuable resource for individuals seeking medical advice. Its ability to understand patient needs and provide tailored recommendations underscores its utility in guiding users toward appropriate medical treatment options.

4.2.2 Spiritual World Applications

A. Language. CoPoet [158] is a collaborative iterative poetry creation model. Users can iteratively request suggestions from CoPoet through natural language instructions, and CoPoet will generate further content based on the user's input. Based on internal feedback (Section 3.3.2), CoPoet refines the creation process or aids users in their creations by assimilating user feedback.

Ippolito et al. [159] have identified barriers to creative writing between AI-driven writing agents and experienced professional writers. They assessed the creative capabilities of the AI writing agent by engaging experts in the field of professional creation from diverse countries, races, and backgrounds. These experts were asked to offer open-ended qualitative feedback on the content generated by the AI writing agent. The authors analyze the evaluation results and proposes lessons that could enhance the creative abilities of large models. Brainstorming emerges as a crucial aspect of creation, and few-shot learning (Section 3.1.3) can be employed to provide prompts for creativity. Task environment information (Section 3.1.2) can help LLM enhance imagination. By providing environmental information to the language model, detailed data can facilitate more effective brainstorming. Additionally, the author suggests that this could also be achieved through methods like RAG (Section 3.2) and APE (Automatic Post-Editing).

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

C. Hearing. ChatMusician [160] adeptly integrates various musical elements, crafting well-structured compositions. By unifying symbolic music understanding and generation tasks, it generates coherent pieces across styles. Utilizing prompt engineering ChatMusician explores musical creation dynamically. It formats tasks with few-shots prompts (Section 3.1.3) to inspire the creation of large models. It also adopts musician role-play prompts (Section 3.1.1) to enhance its perspective.

5 Conclusion

Prompt engineering is gaining increasing significance across various application domains as a crucial technique for optimizing the performance of large language models (LLMs). Drawing inspiration from the division of agent functions, we systematically categorize prompt engineering techniques into four distinct aspects. Based on these categories, we present a comprehensive and up-to-date review of existing prompt engineering researches. Furthermore, we comprehensively review the applications of LLM based on prompt engineering. We aspire for this survey to serve as a comprehensive guide for readers, illuminating the advancements in prompt engineering and offering valuable insights into its practical applications.

References

[1] R. Zhang, Y. Su, B. D. Trisedya, X. Zhao, M. Yang, H. Cheng, and J. Qi, "Autoalign: Fully automatic and effective knowledge graph alignment enabled by large language models," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 6, pp. 2357–2371, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3325484

[2] J. He, Y. Li, Z. Zhai, B. Fang, C. Thorne, C. Druckenbrodt, S. A. Akhondi, and K. Verspoor, "Focused contrastive loss for classification with pre-trained language models," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3047–3061, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3327777

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Amanda, S. Agarwal, A. Herbert-Voss, G. Krueger, H. Tom, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, C. Benjamin, J. Clark, C. Berner, M. Sam, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv: Computation and Language,arXiv: Computation and Language*, May 2020.

[4] G. Abercrombie, A. C. Curry, T. Dinkar, V. Rieser, and Z. Talat, "Mirages. on anthropomorphism in dialogue systems," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, 2023, pp. 4776–4790. [Online]. Available: https://doi.org/10.18653/v1/2023.emnlp-main.290

[5] D. Buren, "Guided scenarios with simulated expert personae: a remarkable strategy to perform cognitive work," *CoRR*, Jun 2023.

[6] M. Zheng, J. Pei, and D. Jurgens, "Is "a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts," *CoRR*, Nov 2023.

[7] J. Park, J. O'Brien, C. Cai, M. Morris, P. Liang, and M. Bernstein, "Generative agents: Interactive simulacra of human behavior," *CoRR*, Apr 2023.

[8] E. Avia and O. Levy, "The turking test: Can language models understand instructions?" *arXiv: Computation and Language,arXiv: Computation and Language*, Oct 2020.

[9] J. Chen, H. Lin, X. Han, and L. Sun, "Benchmarking large language models in retrieval-augmented generation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 16, 2024, pp. 17 754–17 762.

[10] Y. Tang and Y. Yang, "Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries," *ArXiv*, vol. abs/2401.15391, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267312593

[11] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," in *Conference on Empirical Methods in Natural Language Processing*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257505063

[12] X. Cheng, D. Luo, X. Chen, L. Liu, D. Zhao, and R. Yan, "Lift yourself up: Retrieval-augmented text generation with self memory," *ArXiv*, vol. abs/2305.02437, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258479968

[13] Y. Zhou, Z. Liu, J. Jin, J.-Y. Nie, and Z. Dou, "Metacognitive retrieval-augmented large language models," *ArXiv*, vol. abs/2402.11626, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267750831

[14] S. Qiao, Y. Ou, N. Zhang, X. Chen, Y. Yao, S. Deng, C. Tan, F. Huang, and H. Chen, "Reasoning with language model prompting: A survey," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds. Association for Computational Linguistics, 2023, pp. 5368–5393. [Online]. Available: https://doi.org/10.18653/v1/2023.acl-long.294

[15] ——, "Reasoning with language model prompting: A survey," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, A. Rogers, J. L. Boyd-Graber, and N. Okazaki, Eds. Association for Computational Linguistics, 2023, pp. 5368–5393. [Online]. Available: https://doi.org/10.18653/v1/2023.acl-long.294

[16] F. Yu, H. Zhang, P. Tiwari, and B. Wang, "Natural language reasoning, a survey," 2023. [Online]. Available: https://arxiv.org/abs/2303.14725

[17] X. W. Jason Wei, D. Schuurmans, M. Bosma, E. H. hsin Chi, F. Xia, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models," *ArXiv*, vol. abs/2201.11903, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246411621

[18] J. Long, "Large language model guided tree-of-thought," *ArXiv*, vol. abs/2305.08291, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258686311

[19] Y. Yao, Z. Li, and H. Zhao, "Beyond chain-of-thought, effective graph-of-thought reasoning in language models," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:268681737

[20] LangChain-ai, https://www.langchain.com/, 2024-7-10.

[21] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Comput. Surv.*, vol. 55, no. 9, pp. 195:1–195:35, 2023. [Online]. Available: https://doi.org/10.1145/3560815

[22] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," *Trans. Mach. Learn. Res.*, vol. 2022, 2022. [Online]. Available: https://openreview.net/forum?id=yzkSU5zdwD

[23] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *CoRR*, vol. abs/2402.07927, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.07927

[24] H. Li, J. Leung, and Z. Shen, "Towards goal-oriented large language model prompting: A survey," *CoRR*, vol. abs/2401.14043, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.14043

[25] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, E. Grave, Y. LeCun, and T. Scialom, "Augmented language models: a survey," *CoRR*, vol. abs/2302.07842, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.07842

[26] T. Shin, Y. Razeghi, R. L. L. IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Association for Computational Linguistics, 2020, pp. 4222–4235. [Online]. Available: https://doi.org/10.18653/v1/2020.emnlp-main.346

[27] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Jan 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/d19-1250

[28] L. Cui, Y. Wu, J. Liu, S. Yang, and Y. Zhang, "Template-based named entity recognition using BART," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 1835–1845. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-acl.161

[29] T. Vu, B. Lester, N. Constant, R. Al-Rfou', and D. Cer, "Spot: Better frozen model adaptation through soft prompt transfer," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 5039–5059. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.346

[30] Y. Wen, N. Jain, J. Kirchenbauer, M. Goldblum, J. Geiping, and T. Goldstein, "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[31] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: empowering large language models with optimal planning proficiency," *CoRR*, vol. abs/2304.11477, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.11477

[32] W. Zhou, S. Zhang, H. Poon, and M. Chen, "Context-faithful prompting for large language models," 2023. [Online]. Available: https://arxiv.org/abs/2303.11315

[33] S. Li, X. Ning, L. Wang, T. Liu, X. Shi, S. Yan, G. Dai, H. Yang, and Y. Wang, "Evaluating quantized large language models," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=DKKg5EFAFr

[34] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang, X. Sun, L. Li, and Z. Sui, "A survey on in-context learning," 2024. [Online]. Available: https://arxiv.org/abs/2301.00234

[35] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen, "A survey on large language model based autonomous agents," *Frontiers Comput. Sci.*, vol. 18, no. 6, p. 186345, 2024. [Online]. Available: https://doi.org/10.1007/s11704-024-40231-1

[36] K. Yang, J. Liu, J. Wu, C. Yang, Y. R. Fung, S. Li, Z. Huang, X. Cao, X. Wang, Y. Wang, H. Ji, and C. Zhai, "If LLM is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents," *CoRR*, vol. abs/2401.00812, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.00812

[37] D. Roy, X. Zhang, R. Bhave, C. Bansal, P. H. B. Las-Casas, R. Fonseca, and S. Rajmohan, "Exploring llm-based agents for root cause analysis," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, FSE 2024, Porto de Galinhas, Brazil, July 15-19, 2024*, M. d'Amorim, Ed. ACM, 2024, pp. 208–219. [Online]. Available: https://doi.org/10.1145/3663529.3663841

[38] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," *arXiv: Computation and Language,arXiv: Computation and Language*, Feb 2021.

[39] X. Xu, C. Tao, T. Shen, C. Xu, H. Xu, G. Long, and J.-g. Lou, "Re-reading improves reasoning in language models," *CoRR*, Sep 2023.

[40] Z. Wang, S. Cai, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," *CoRR*, vol. abs/2302.01560, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.01560

[41] X. Zhu, Y. Chen, H. Tian, C. Tao, W. Su, C. Yang, G. Huang, B. Li, L. Lu, X. Wang, Y. Qiao, Z. Zhang, and J. Dai, "Ghost in the minecraft: Generally capable agents for open-world environments via large language models with text-based knowledge and memory," 2023. [Online]. Available: https://arxiv.org/abs/2305.17144

[42] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *ArXiv*, vol. abs/2205.11916, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:249017743

[43] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[44] R. Logan IV, I. Balazevic, E. Wallace, F. Petroni, S. Singh, and S. Riedel, "Cutting down on prompts and parameters: Simple few-shot learning with language models," in *Findings of the Association for Computational Linguistics: ACL 2022*, Jan 2022. [Online]. Available: http://dx.doi.org/10.18653/v1/2022.findings-acl.222

[45] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What makes good in-context examples for gpt-3?" in *Proceedings of Deep Learning Inside Out: The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures, DeeLIO@ACL 2022, Dublin, Ireland and Online, May 27, 2022*, E. Agirre, M. Apidianaki, and I. Vulic, Eds. Association for Computational Linguistics, 2022, pp. 100–114. [Online]. Available: https://doi.org/10.18653/v1/2022.deelio-1.10

[46] H. Su, J. Kasai, C. H. Wu, W. Shi, T. Wang, J. Xin, R. Zhang, M. Ostendorf, L. Zettlemoyer, N. A. Smith, and T. Yu, "Selective annotation makes language models better few-shot learners," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=qY1hlv7gwg

[47] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 423–438, 2020. [Online]. Available: https://doi.org/10.1162/tacl_a_00324

[48] Y. Lu, M. Bartolo, A. Moore, S. Riedel, and P. Stenetorp, "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 (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, S. Muresan, P. Nakov, and A. Villavicencio, Eds. Association for Computational Linguistics, 2022, pp. 8086–8098. [Online]. Available: https://doi.org/10.18653/v1/2022.acl-long.556

[49] F. Nie, M. Chen, Z. Zhang, and X. Cheng, "Improving few-shot performance of language models via nearest neighbor calibration," *CoRR*, vol. abs/2212.02216, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2212.02216

[50] Y. Fei, Y. Hou, Z. Chen, and A. Bosselut, "Mitigating label biases for in-context learning," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 14 014–14 031. [Online]. Available: https://aclanthology.org/2023.acl-long.783

[51] H. Ma, C. Zhang, Y. Bian, L. Liu, Z. Zhang, P. Zhao, S. Zhang, H. Fu, Q. Hu, and B. Wu, "Fairness-guided few-shot prompting for large language models," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/8678da90126aa58326b2fc0254b33a8c-Abstract-Conference.html

[52] Y. Reif and R. Schwartz, "Beyond performance: Quantifying and mitigating label bias in LLMs," in *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 6784–6798. [Online]. Available: https://aclanthology.org/2024.naacl-long.378

[53] Z. Han, Y. Hao, L. Dong, Y. Sun, and F. Wei, "Prototypical calibration for few-shot learning of language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=nUsI9lFADUF

[54] L. Hu, Z. Liu, Z. Zhao, L. Hou, L. Nie, and J. Li, "A survey of knowledge enhanced pre-trained language models," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 4, pp. 1413–1430, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2023.3310002

[55] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems 33: Annual Conference*

on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

[56] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3580–3599, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2024.3352100

[57] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 7, pp. 3091–3110, 2024. [Online]. Available: https://doi.org/10.1109/TKDE.2024.3360454

[58] S. Barnett, S. Kurniawan, S. Thudumu, Z. Brannelly, and M. Abdelrazek, "Seven failure points when engineering a retrieval augmented generation system," in *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN 2024, Lisbon, Portugal, April 14-15, 2024*, J. Cleland-Huang, J. Bosch, H. Muccini, and G. A. Lewis, Eds. ACM, 2024, pp. 194–199. [Online]. Available: https://doi.org/10.1145/3644815.3644945

[59] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," *ArXiv*, vol. abs/2305.15294, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258866037

[60] W. Yu, H. Zhang, X. Pan, K. Ma, H. Wang, and D. Yu, "Chain-of-note: Enhancing robustness in retrieval-augmented language models," *ArXiv*, vol. abs/2311.09210, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265212816

[61] E. Melz, "Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation," *ArXiv*, vol. abs/2311.04177, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265043634

[62] F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, "The power of noise: Redefining retrieval for rag systems," *ArXiv*, vol. abs/2401.14887, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267301416

[63] J. Weston and S. Sukhbaatar, "System 2 attention (is something you might need too)," *ArXiv*, vol. abs/2311.11829, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:265295357

[64] J. T. Austin and J. B. Vancouver, "Goal constructs in psychology: Structure, process, and content." *Psychological Bulletin*, vol. 120, pp. 338–375, 1996. [Online]. Available: https://api.semanticscholar.org/CorpusID:197656249

[65] D. Zhou, N. Scharli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. H. hsin Chi, "Least-to-most prompting enables complex reasoning in large language models," *ArXiv*, vol. abs/2205.10625, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248986239

[66] Z. Zhang, A. Zhang, M. Li, and A. J. Smola, "Automatic chain of thought prompting in large language models," *ArXiv*, vol. abs/2210.03493, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252762275

[67] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201646309

[68] S. Diao, P. Wang, Y. Lin, and T. Zhang, "Active prompting with chain-of-thought for large language models," *ArXiv*, vol. abs/2302.12246, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257102707

[69] Q. Lyu, S. Havaldar, A. Stein, L. Zhang, D. Rao, E. Wong, M. Apidianaki, and C. Callison-Burch, "Faithful chain-of-thought reasoning," *ArXiv*, vol. abs/2301.13379, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:256416127

[70] T. Liu, Q. Guo, Y. Yang, X. Hu, Y. Zhang, X. Qiu, and Z. Zhang, "Plan, verify and switch: Integrated reasoning with diverse x-of-thoughts," *ArXiv*, vol. abs/2310.14628, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264426101

[71] R. Liu, J. Wei, S. Gu, T.-Y. Wu, S. Vosoughi, C. Cui, D. Zhou, and A. Dai, "Mind's eye: Grounded language model reasoning through simulation," Oct 2022.

[72] L. Li, J. Xu, Q. Dong, C. Zheng, Q. Liu, L. Kong, and X. Sun, "Can language models understand physical concepts?"

[73] Y. Wang, J. Duan, D. Fox, and S. Srinivasa, "Newton: Are large language models capable of physical reasoning?" Oct 2023.

[74] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012. [Online]. Available: http://dx.doi.org/10.1109/iros.2012.6386109

[75] I. Drori, S. Tran, R. Wang, N. Cheng, K. Liu, L. Tang, E. Ke, N. Singh, T. L. Patti, J. Lynch, A. Shporer, N. Verma, E. Wu, and G. Strang, "A neural network solves and generates mathematics problems by program synthesis: Calculus, differential equations, linear algebra, and more," *CoRR*, vol. abs/2112.15594, 2021. [Online]. Available: https://arxiv.org/abs/2112.15594

[76] W. Chen, X. Ma, X. Wang, and W. W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," *CoRR*, vol. abs/2211.12588, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.12588

[77] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[78] Z. Cheng, T. Xie, P. Shi, C. Li, R. Nadkarni, Y. Hu, C. Xiong, D. Radev, M. Ostendorf, L. Zettlemoyer, N. Smith, and T. Yu, "Binding language models in symbolic languages," Oct 2022.

[79] Y. Wu, A. Q. Jiang, W. Li, M. N. Rabe, C. Staats, M. Jamnik, and C. Szegedy, "Autoformalization with large language models," in *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., 2022. [Online]. Available: http://papers.nips.cc/paper_files/paper/2022/hash/d0c6bc641a56bebee9d985b937307367-Abstract-Conference.html

[80] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," Nov 2022.

[81] W. Chen, X. Ma, X. Wang, and W. Cohen, "Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks," Nov 2022.

[82] X. Luo, Q. Zhu, Z. Zhang, L. Qin, X. Wang, Q. Yang, D. Xu, and W. Che, "Multipot: Multilingual program of thoughts harnesses multiple programming languages," *CoRR*, vol. abs/2402.10691, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.10691

[83] A. Zeng, A. Wong, S. Welker, K. Choromanski, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence, "Socratic models: Composing zero-shot multimodal reasoning with language."

[84] A. Parisi, Y. Zhao, and N. Fiedel, "Talm: Tool augmented language models," May 2022.

[85] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, E. Hambro, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language models can teach themselves to use tools," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/d842425e4bf79ba039352da0f658a906-Abstract-Conference.html

[86] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez, and R. Stojnic, "Galactica: A large language model for science," *CoRR*, vol. abs/2211.09085, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2211.09085

[87] E. Karpas, O. Abend, Y. Belinkov, B. Lenz, O. Lieber, N. Ratner, Y. Shoham, H. Bata, Y. Levine, K. Leyton-Brown, D. Muhlgay, N. Rozen, E. Schwartz, G. Shachaf, S. Shalev-Shwartz, A. Shashua, and M. Tenenholtz, "Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning," May 2022.

[88] B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer, and M. Ribeiro, "Art: Automatic multi-step reasoning and tool-use for large language models," Mar 2023.

[89] P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. Wu, S.-C. Zhu, and J. Gao, "Chameleon: Plug-and-play compositional reasoning with large language models."

[90] S. G. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large language model connected with massive apis," *CoRR*, vol. abs/2305.15334, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.15334

[91] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023.

[92] Q. Tang, Z. Deng, H. Lin, X. Han, Q. Liang, and L. Sun, "Toolalpaca: Generalized tool learning for language models with 3000 simulated cases," *CoRR*, vol. abs/2306.05301, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.05301

[93] J. Ruan, Y. Chen, B. Zhang, Z. Xu, T. Bao, G. Du, S. Shi, H. Mao, X. Zeng, and R. Zhao, "TPTU: task planning and tool usage of large language model-based AI agents," *CoRR*, vol. abs/2308.03427, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.03427

[94] Y. Kong, J. Ruan, Y. Chen, B. Zhang, T. Bao, S. Shi, G. Du, X. Hu, H. Mao, Z. Li, X. Zeng, and R. Zhao, "Tptu-v2: Boosting task planning and tool usage of large language model-based agents in real-world systems," *CoRR*, vol. abs/2311.11315, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2311.11315

[95] Y. Liang, C. Wu, T. Song, W. Wu, Y. Xia, Y. Liu, Y. Ou, S. Lu, L. Ji, S. Mao, Y. Wang, L. Shou, M. Gong, and N. Duan, "Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis," *CoRR*, vol. abs/2303.16434, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2303.16434

[96] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Welleck, B. P. Majumder, S. Gupta, A. Yazdanbakhsh, and P. Clark, "Self-refine: Iterative refinement with self-feedback," *ArXiv*, vol. abs/2303.17651, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257900871

[97] J. Lu, W. Zhong, W. Huang, Y. Wang, F. Mi, B. Wang, W. Wang, L. Shang, and Q. Liu, "Self: Self-evolution with language feedback," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263334155

[98] W. Zhang, Y. Shen, L. Wu, Q. Peng, J. Wang, Y. T. Zhuang, and W. Lu, "Self-contrast: Better reflection through inconsistent solving perspectives," *ArXiv*, vol. abs/2401.02009, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:266755862

[99] S. Wang, S. Li, T. Sun, J. Fu, Q. Cheng, J. Ye, J. Ye, X. Qiu, and X. Huang, "Llm can achieve self-regulation via hyperparameter aware generation," *ArXiv*, vol. abs/2402.11251, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267750641

[100] X. Wang, Z. Wang, J. Liu, Y. Chen, L. Yuan, H. Peng, and H. Ji, "Mint: Evaluating llms in multi-turn interaction with tools and language feedback," *ArXiv*, vol. abs/2309.10691, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:262053695

[101] H. Huang, Y. Qu, J. Liu, M. Yang, and T. Zhao, "Self-evaluation of large language model based on glass-box features," 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:268264196

[102] J. Huang, X. Chen, S. Mishra, H. Zheng, A. Yu, X. Song, and D. Zhou, "Large language models cannot self-correct reasoning yet," Oct 2023.

[103] W. Xu, G. Zhu, X. Zhao, L. Pan, L. Li, and W. Y. Wang, "Perils of self-feedback: Self-bias amplifies in large language models," *ArXiv*, vol. abs/2402.11436, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:267751249

[104] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, "Critic: Large language models can self-correct with tool-interactive critiquing," *ArXiv*, vol. abs/2305.11738, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258823123

[105] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 2021, pp. 12 697–12 706. [Online]. Available: http://proceedings.mlr.press/v139/zhao21c.html

[106] C. Si, Z. Gan, Z. Yang, S. Wang, J. Wang, J. Boyd-Graber, and L. Wang, "Prompting gpt-3 to be reliable," Oct 2022.

[107] P. Liang, R. Bommasani, T. Lee, and T. et. al, "Holistic evaluation of language models," Nov 2022.

[108] X. Ye and G. Durrett, "The unreliability of explanations in few-shot in-context learning," May 2022.

[109] O. Shaikh, H. Zhang, W. Held, M. Bernstein, and D. Yang, "On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning," Dec 2022.

[110] S. Lin, J. Hilton, and O. Evans, "Truthfulqa: Measuring how models mimic human falsehoods," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jan 2022. [Online]. Available: http://dx.doi.org/10.18653/v1/2022.acl-long.229

[111] T. Schick and H. Schütze, "Exploiting cloze-questions for few-shot text classification and natural language inference," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Jan 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2021.eacl-main.20

[112] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Jan 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2021.emnlp-main.243

[113] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen, "Making large language models better reasoners with step-aware verifier."

[114] S. Pitis, M. Zhang, A. Wang, and J. Ba, "Boosted prompt ensembles for large language models," Apr 2023.

[115] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[116] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou, and W. Chen, "On the advance of making language models better reasoners," *arXiv preprint arXiv:2206.02336*, 2022.

[117] S. Arora, A. Narayan, M. F. Chen, L. J. Orr, N. Guha, K. Bhatia, I. Chami, and C. Ré, "Ask me anything: A simple strategy for prompting language models," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/forum?id=bhUPJnS2g0X

[118] B. Hou, J. O'Connor, J. Andreas, S. Chang, and Y. Zhang, "Promptboosting: Black-box text classification with ten forward passes," Dec 2022.

[119] C. Zhang, L. Liu, C. Wang, X. Sun, H. Wang, J. Wang, and M. Cai, "Prefer: Prompt ensemble learning via feedback-reflect-refine," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 17, 2024, pp. 19 525–19 532.

[120] T. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," *arXiv: Computation and Language,arXiv: Computation and Language*, Feb 2021.

[121] J. Allingham, J. Ren, M. Dusenberry, J. Liu, X. Gu, Y. Cui, D. Tran, and B. Lakshminarayanan, "A simple zero-shot prompt weighting technique to improve prompt ensembling in text-image models," Feb 2023.

[122] N. Meade, E. Poole-Dayan, and S. Reddy, "An empirical survey of the effectiveness of debiasing techniques for pre-trained language models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jan 2022. [Online]. Available: http://dx.doi.org/10.18653/v1/2022.acl-long.132

[123] M. Nadeem, A. Bethke, and S. Reddy, "Stereoset: Measuring stereotypical bias in pretrained language models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Jan 2021. [Online]. Available: http://dx.doi.org/10.18653/v1/2021.acl-long.416

[124] Z. Tang, K. Zhou, J. Li, Y. Ding, P. Wang, B. Yan, and M. Zhang, "Cmd: a framework for context-aware model self-detoxification," 2024.

[125] R. Bowman, O. Cooney, J. W. Newbold, A. Thieme, L. Clark, G. Doherty, and B. Cowan, "Exploring how politeness impacts the user experience of chatbots for mental health support," *International Journal of Human-Computer Studies*, vol. 184, p. 103181, 2024.

[126] A. Glaese, N. McAleese, M. Trebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. J. Chadwick, P. Thacker, L. Campbell-Gillingham, J. Uesato, P. Huang, R. Comanescu, F. Yang, A. See, S. Dathathri, R. Greig, C. Chen, D. Fritz, J. S. Elias, R. Green, S. Mokrá, N. Fernando, B. Wu, R. Foley, S. Young, I. Gabriel, W. Isaac, J. Mellor, D. Hassabis, K. Kavukcuoglu, L. A. Hendricks, and G. Irving, "Improving alignment of dialogue agents via targeted human judgements," *CoRR*, vol. abs/2209.14375, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2209.14375

[127] OpenAI, https://openai.com/index/chatgpt/, 2024-5-10.

[128] OpenAI, J. Achiam, S. Adler, and S. Agarwal, "Gpt-4 technical report," 2024.

[129] Microsoft, https://copilot.microsoft.com/, 2024-5-10.

[130] "The claude 3 model family: Opus, sonnet, haiku." [Online]. Available: https://api.semanticscholar.org/CorpusID:268232499

[131] L. Gao, X. Ma, J. Lin, and J. Callan, "Precise zero-shot dense retrieval without relevance labels," Dec 2022.

[132] L. Wang, N. Yang, and F. Wei, "Query2doc: Query expansion with large language models," Mar 2023.

[133] R. Jagerman, H. Zhuang, Z. Qin, X. Wang, and M. Bendersky, "Query expansion by prompting large language models," May 2023.

[134] M. Li, H. Zhuang, K. Hui, Z. Qin, J. Lin, R. Jagerman, X. Wang, and M. Bendersky, "Generate, filter, and fuse: Query expansion via multi-step keyword generation for zero-shot neural rankers," Nov 2023.

[135] Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li, "Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, R. Barzilay and M. Kan, Eds. Association for Computational Linguistics, 2017, pp. 496–505. [Online]. Available: https://doi.org/10.18653/v1/P17-1046

[136] Y. Arens, C. A. Knoblock, and W.-M. Shen, "Query reformulation for dynamic information integration," *Journal of Intelligent Information Systems*, p. 99–130, Jun 1996. [Online]. Available: http://dx.doi.org/10.1007/bf00122124

[137] J. Huang and E. N. Efthimiadis, "Analyzing and evaluating query reformulation strategies in web search logs," in *Proceedings of the 18th ACM conference on Information and knowledge management*, Nov 2009.

[Online]. Available: http://dx.doi.org/10.1145/1645953.1645966
[138] R. F. Nogueira, Z. Jiang, R. Pradeep, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," in *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, ser. Findings of ACL, T. Cohn, Y. He, and Y. Liu, Eds., vol. EMNLP 2020. Association for Computational Linguistics, 2020, pp. 708–718. [Online]. Available: https://doi.org/10.18653/v1/2020.findings-emnlp.63
[139] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. van den Driessche, J. Lespiau, B. Damoc, A. Clark, D. de Las Casas, A. Guy, J. Menick, R. Ring, T. Hennigan, S. Huang, L. Maggiore, C. Jones, A. Cassirer, A. Brock, M. Paganini, G. Irving, O. Vinyals, S. Osindero, K. Simonyan, J. W. Rae, E. Elsen, and L. Sifre, "Improving language models by retrieving from trillions of tokens," in *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 2022, pp. 2206–2240. [Online]. Available: https://proceedings.mlr.press/v162/borgeaud22a.html
[140] N. Ziems, W. Yu, Z. Zhang, and M. Jiang, "Large language models are built-in autoregressive search engines."
[141] H. Liu, Y. Liao, Y. Meng, and Y. Wang, "Xiezhi: Chinese law large language model," https://github.com/LiuHC0428/LAW_GPT, 2023.
[142] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl, P. Payne, M. Seneviratne, P. Gamble, C. Kelly, N. Scharli, A. Chowdhery, P. Mansfield, B. A. y Arcas, D. Webster, G. S. Corrado, Y. Matias, K. Chou, J. Gottweis, N. Tomasev, Y. Liu, A. Rajkomar, J. Barral, C. Semturs, A. Karthikesalingam, and V. Natarajan, "Large language models encode clinical knowledge," 2022.
[143] K. M. Yoo, D. Park, J. Kang, S. Lee, and W. Park, "Gpt3mix: Leveraging large-scale language models for text augmentation," in *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, M. Moens, X. Huang, L. Specia, and S. W. Yih, Eds. Association for Computational Linguistics, 2021, pp. 2225–2239. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-emnlp.192
[144] L. H. Bonifacio, H. Q. Abonizio, M. Fadaee, and R. F. Nogueira, "Inpars: Unsupervised dataset generation for information retrieval," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, Eds. ACM, 2022, pp. 2387–2392. [Online]. Available: https://doi.org/10.1145/3477495.3531863
[145] V. Jeronymo, L. H. Bonifacio, H. Q. Abonizio, M. Fadaee, R. de Alencar Lotufo, J. Zavrel, and R. F. Nogueira, "Inpars-v2: Large language models as efficient dataset generators for information retrieval," *CoRR*, vol. abs/2301.01820, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2301.01820
[146] H. Dai, Z. Liu, W. Liao, X. Huang, Z. Wu, L. Zhao, W. Liu, N. Liu, S. Li, D. Zhu, H. Cai, Q. Li, D. Shen, T. Liu, and X. Li, "Chataug: Leveraging chatgpt for text data augmentation," *CoRR*, vol. abs/2302.13007, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.13007
[147] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg, and G. Mann, "Bloomberggpt: A large language model for finance," 2023.
[148] H. Yang, X.-Y. Liu, and C. D. Wang, "Fingpt: Open-source financial large language models," 2023.
[149] R. K. Luu and M. J. Buehler, "Bioinspiredllm: Conversational large language model for the mechanics of biological and bio-inspired materials," 2023.
[150] S. Liu, J. Wang, Y. Yang, C. Wang, L. Liu, H. Guo, and C. Xiao, "Chatgpt-powered conversational drug editing using retrieval and domain feedback," 2023.
[151] A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White, and P. Schwaller, "Chemcrow: Augmenting large-language models with chemistry tools," 2023.
[152] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. [Online]. Available: https://openreview.net/pdf?id=iaYcJKpY2B_
[153] Q. Zheng, X. Xia, X. Zou, Y. Dong, S. Wang, Y. Xue, L. Shen, Z. Wang, A. Wang, Y. Li, T. Su, Z. Yang, and J. Tang, "Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, A. K. Singh, Y. Sun, L. Akoglu, D. Gunopulos, X. Yan, R. Kumar, F. Ozcan, and J. Ye, Eds. ACM, 2023, pp. 5673–5684. [Online]. Available: https://doi.org/10.1145/3580305.3599790
[154] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "Palm-e: An embodied multimodal language model," 2023.
[155] W. Yiquan, L. Yuhang, L. Yifei, L. Ang, Z. Siying, and K. Kun, "wisdominterrogatory," available at GitHub. [Online]. Available: https://github.com/zhihaiLLM/wisdomInterrogatory
[156] S. Yue, W. Chen, S. Wang, B. Li, C. Shen, S. Liu, Y. Zhou, Y. Xiao, S. Yun, X. Huang, and Z. Wei, "Disc-lawllm: Fine-tuning large language models for intelligent legal services," 2023.
[157] Y. Li, Z. Li, K. Zhang, R. Dan, S. Jiang, and Y. Zhang, "Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge," 2023.
[158] T. Chakrabarty, V. Padmakumar, and H. He, "Help me write a poem: Instruction tuning as a vehicle for collaborative poetry writing," 2022.
[159] D. Ippolito, A. Yuan, A. Coenen, and S. Burnam, "Creative writing with an ai-powered writing assistant: Perspectives from professional writers," 2022.
[160] R. Yuan, H. Lin, Y. Wang, Z. Tian, S. Wu, T. Shen, G. Zhang, Y. Wu, C. Liu, Z. Zhou, Z. Ma, L. Xue, Z. Wang, Q. Liu, T. Zheng, Y. Li, Y. Ma, Y. Liang, X. Chi, R. Liu, Z. Wang, P. Li, J. Wu, C. Lin, Q. Liu, T. Jiang, W. Huang, W. Chen, E. Benetos, J. Fu, G. Xia, R. Dannenberg, W. Xue, S. Kang, and Y. Guo, "Chatmusician: Understanding and generating music intrinsically with llm," 2024.