Prompt Programming and Prompting Language in Large Language Models: A Survey

Abstract

Prompt programming, a transformative approach within artificial intelligence (AI) and natural language processing (NLP), leverages large language models (LLMs) to perform diverse tasks through strategically crafted inputs. This survey explores the paradigm shift prompted by this technique, highlighting its applications in domains such as healthcare, education, and software engineering. The integration of prompt programming enhances LLM adaptability, enabling complex problem-solving and tool generation, while addressing challenges like prompt sensitivity and reliability. Key innovations include advanced prompting techniques, hybrid frameworks, and automated strategies that optimize model performance. Despite its potential, prompt programming faces limitations in scalability, bias, and ethical concerns, necessitating ongoing research to refine methodologies and ensure responsible AI deployment. By examining the historical development and core concepts of LLMs, this survey underscores the significance of prompt programming in advancing AI technologies. Future research directions aim to enhance LLM efficiency, explore ethical guidelines, and improve grounding in multi-modal tasks, ensuring the continued evolution and impact of prompt programming across diverse applications.

1 Introduction

1.1 Overview of Prompt Programming

Prompt programming has emerged as a transformative paradigm in artificial intelligence (AI) and natural language processing (NLP), particularly with the rise of large language models (LLMs). This approach utilizes pre-trained language models (PLMs) to execute a variety of tasks by formulating specific natural language inputs, thereby guiding these models to produce desired outputs [1]. It enhances the adaptability and accuracy of LLMs across diverse applications, including psychotherapy, clinical practice, and complex combinatorial problems.

The strategic application of prompts allows LLMs to address structured tasks, such as extracting structured data from unstructured text, a challenge that remains despite advancements in NLP [2]. Furthermore, prompt programming supports the generation of accurate and interpretable code models through methods like Inductive-Bias Learning (IBL), which combines in-context learning with code generation techniques, optimizing the quality of LLM-produced code.

However, prompt programming encounters challenges, including the instability of implicit reasoning and limitations related to API availability when using LLM tools [3]. Moreover, LLMs may generate inaccurate content despite being trained on factual data, highlighting the necessity for refined prompting techniques to ensure reliability in AI outputs [4].

In education, prompt programming aids educators in adapting pedagogical approaches to leverage LLM capabilities while addressing associated challenges [5]. As LLM deployment in real-world applications increases, the significance of prompt programming in enhancing AI safety and functionality becomes evident, positioning it as an essential tool in the evolution of AI technologies [6].

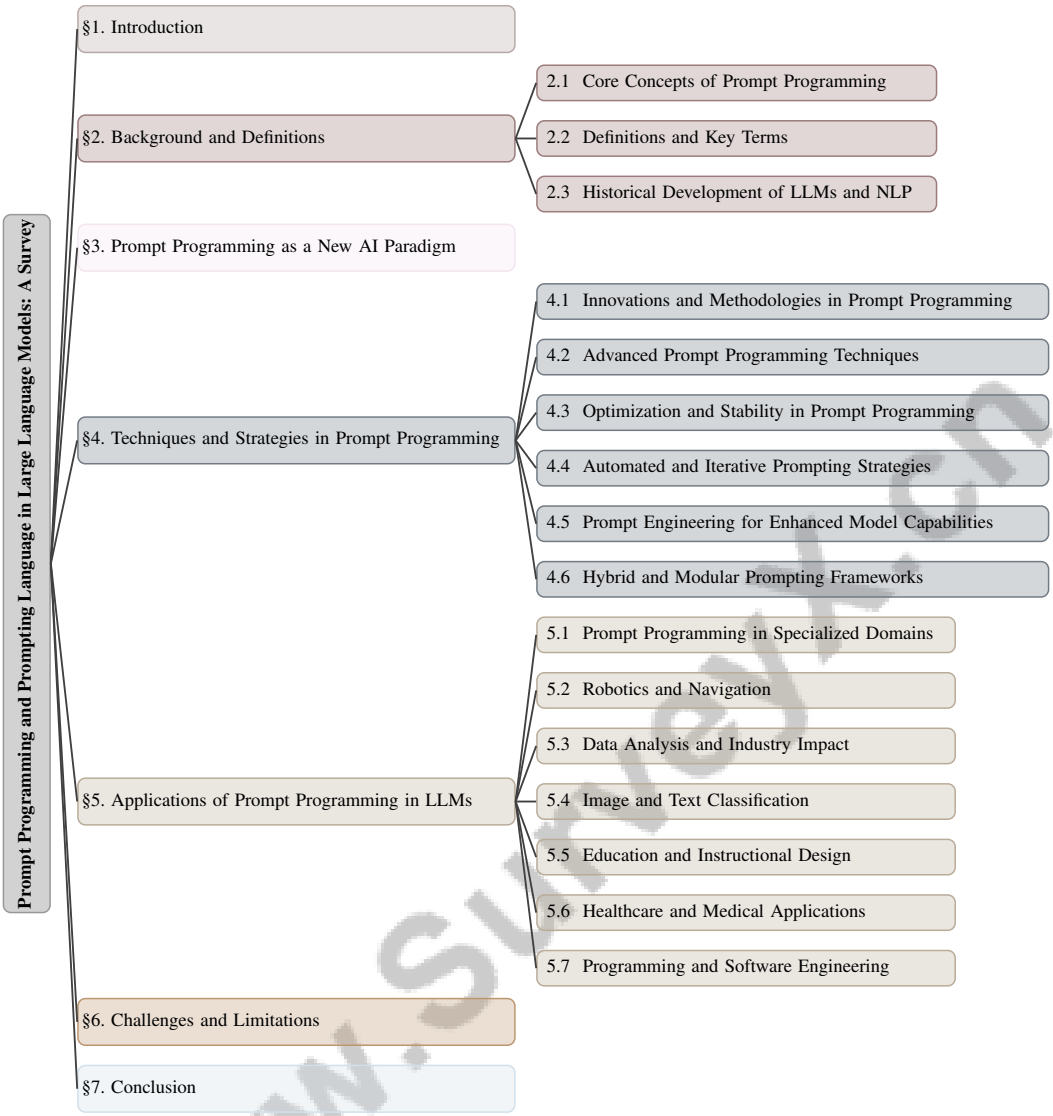


Figure 1: chapter structure

1.2 Significance in AI and NLP

Prompt programming is pivotal in advancing AI and NLP technologies, significantly enhancing the functionality and precision of LLMs. By enabling LLMs to act as tool creators, prompt programming facilitates the resolution of complex problems with greater accuracy and flexibility. This is achieved through the models' ability to generate and modify tools tailored to specific tasks, thus expanding their problem-solving capabilities [3]. The integration of verification techniques with LLMs further refines AI outputs, addressing the brittleness of traditional prompting methods and establishing a more stable framework for AI interactions [7]. Such stability is crucial in mitigating variability in model predictions that can result from minor prompt modifications [8].

In education, the advent of LLMs necessitates a reevaluation of pedagogical strategies and assessment practices, given their potential to enhance learning while posing risks to academic integrity [5]. The development of metaprompts, which allow models to autonomously generate their own prompts, marks an innovative advancement in prompt programming, fostering more robust engagement with AI tools [9]. Addressing knowledge gaps regarding AI tool effectiveness and challenges in educational contexts is essential for the responsible integration of LLMs, ensuring their adoption aligns with educational objectives without compromising academic standards [10].

AI-generated, for reference only.

Prompt programming also significantly enhances machine translation performance, illustrating its importance in managing complexities inherent in NLP tasks [11]. By decoupling the decomposition and solving processes in reasoning tasks, it enables the creation of smaller, more efficient models, optimizing performance [12]. The inadequacy of current prompting techniques in addressing combinatorial problems underscores the need for novel strategies to improve performance, emphasizing the critical role of prompt programming in advancing AI and NLP technologies [13].

Additionally, utilizing SQL as a querying interface to extract structured relations from LLMs demonstrates the precision and expressiveness achieved through prompt programming, surpassing traditional natural language prompts [2]. Establishing benchmarks for generating labeled training data, such as those for classifying Big Five personality traits, is crucial for advancing NLP research and highlights the broad applicability of prompt programming [14]. The introduction of benchmarks to evaluate 'according-to prompting' effectiveness further addresses the grounding of LLM outputs, mitigating hallucination issues and enhancing the reliability of AI-generated content [4].

Thus, prompt programming represents a transformative paradigm in AI and NLP, presenting unique challenges and opportunities for innovation. As the field evolves, integrating prompt programming into AI and NLP workflows will enhance the capabilities and applications of these technologies, driving advancements and opening new avenues for exploration [15].

1.3 Structure of the Survey

The survey is structured to offer a comprehensive exploration of prompt programming and prompting language within large language models (LLMs), reflecting its significance in the evolving landscape of artificial intelligence (AI) and natural language processing (NLP). It begins with an introduction to prompt programming, establishing its transformative role in AI and NLP. A detailed background section follows, elucidating core concepts, definitions, and the historical development of LLMs and NLP, providing foundational insights into prompt programming's emergence.

Subsequently, the survey positions prompt programming as a new AI paradigm, examining its leverage of NLP as a form of programming and contrasting it with traditional programming approaches. This section highlights the paradigm shift prompted by prompt programming in AI methodologies.

The survey then explores various techniques and strategies in prompt programming, emphasizing innovations, advanced techniques, optimization methods, and the role of prompt engineering in enhancing LLM capabilities. It includes a detailed examination of automated and iterative prompting strategies, as well as hybrid and modular prompting frameworks, elucidating the methodologies underlying prompt programming. The evolution of prompt engineering with generative pre-trained models like GPT-4 is discussed, emphasizing prompt programming's distinct nature compared to traditional software development. Insights from interviews with developers reveal challenges in crafting effective prompts and the potential for new tools to enhance this process. Additionally, it addresses prompt integration into user interfaces via frameworks like Prompt Middleware, facilitating user interaction with LLMs, and introduces AUTOPROMPT, an automated approach for generating prompts that improves knowledge elicitation accuracy from language models [16, 15, 17].

Following this, the survey reviews diverse applications of prompt programming in LLMs across specialized domains, including robotics, data analysis, image and text classification, education, healthcare, and software engineering, illustrating the practical impact of prompt programming on industry and research.

The challenges and limitations of prompt programming are thoroughly examined, highlighting issues such as prompt sensitivity—where slight wording changes can drastically alter model responses—reliability concerns from the difficulty in developing consistent mental models of model behavior, the labor-intensive nature of manual prompt crafting, inherent biases reflected in prompts, and scalability problems when applying effective prompts across various models and contexts [15, 18, 17, 19, 20]. The survey concludes by summarizing key findings and reflecting on future research directions, emphasizing the importance of continued exploration in the field of prompt programming and prompting language.The following sections are organized as shown in Figure 1.

2 Background and Definitions

2.1 Core Concepts of Prompt Programming

Prompt programming optimizes large language models (LLMs) by crafting input prompts that enhance task performance. This method requires analyzing input parameters that affect LLM behavior, crucial for refining strategies and understanding outputs [4]. Central to this approach is the use of templates or prompt patterns, which provide a structured framework to assist users, particularly non-experts, in creating effective prompts for efficient task execution [21]. Task complexity quantification is vital for evaluating LLM performance, highlighting the need for robust benchmarking [22]. The limitations of current conversational recommendation systems (CRSs) in managing sub-tasks underscore prompt programming's role in enhancing personalized recommendations [5].

Advanced strategies like Self-Guiding Exploration (SGE) allow autonomous generation of multiple thought trajectories for combinatorial problems, breaking them into subtasks for refined solutions [21]. Frameworks such as Prompt-Aided Language Models (PAL) enhance problem-solving by enabling LLMs to produce intermediate reasoning steps, executed by an external interpreter to reach final answers [22]. In educational technology, prompt-based strategies facilitate generative AI tools' effective use for code generation and error resolution [5]. Exploring methodologies and applications of synthetic data generated by LLMs highlights prompt programming's diverse potential [6].

Prompt programming is crucial in dialogue systems, where structured prompts guide LLMs to generate contextually relevant responses in mixed-initiative dialogue tasks. Techniques like Self-Explanation Prompting (SEP) enhance dialogue generation by prompting LLMs to explain each utterance before answering, improving coherence and relevance [21]. Integrating advanced natural language generation (NLG) models with prompt programming enriches learning experiences, underscoring LLMs' educational tool role. However, LLMs' susceptibility to prompt hacking, which can manipulate outputs, necessitates robust strategies [4]. Crafting prompts that generalize across various math word problems, especially in few-shot contexts, remains a challenge, illustrating the demand for continued innovation [22].

2.2 Definitions and Key Terms

In AI and NLP, several foundational terms are essential for understanding and applying prompt programming within LLMs. 'Prompt programming' involves the strategic design and deployment of prompts to guide LLM outputs effectively, emphasizing task specification over example-based learning [23]. This approach facilitates nuanced interactions with LLMs, marking a paradigm shift in task execution. 'Prompting language' refers to structured language use to enhance LLM performance across tasks, including structured data extraction from unstructured text via SQL [2]. This includes using 'situated natural language explanations' (NLEs) for contextually appropriate outputs [24].

'Large language models' (LLMs) are sophisticated AI systems trained on extensive datasets to produce human-like text, capable of executing a broad range of language-based tasks [6]. These models are critical in applications requiring 'functional correctness' and 'code synthesis', where the quality and accuracy of generated code are evaluated [25]. 'Natural language processing' (NLP) focuses on enabling machines to process and analyze vast amounts of natural language data, facilitating effective human-machine communication. Techniques like 'compositional zero-shot learning' (CZSL) and 'language-informed distributions' enhance LLM capabilities, especially in visual primitives contexts [21].

'AI paradigms' encompass the frameworks and methodologies guiding AI technology development and application, including LLMs' integration into diverse domains. These paradigms address challenges like the 'fact selection problem', which involves determining the optimal facts to include in prompts to improve LLM performance [26]. 'Multi-modal Prompt Learning (MaPLe)' fine-tunes a pre-trained vision-language model by learning prompts in both vision and language branches, emphasizing visual and textual data intersection in prompt programming [22]. 'Hallucination' refers to LLMs generating content not grounded in input data, necessitating robust 'prompting strategies' to ensure AI outputs' accuracy and reliability [4]. Understanding these key terms is essential for exploring prompt programming's complexities and innovations in AI and NLP applications.

2.3 Historical Development of LLMs and NLP

The historical development of LLMs and NLP has been marked by significant methodological advancements and ongoing AI capability refinement. Initially reliant on rule-based systems and statistical models, NLP was limited in understanding and generating human-like text. The emergence of LLMs marked a paradigm shift, driven by large datasets and increased computational power, enabling models to exhibit emergent abilities unreplicable by smaller models [27]. This transition necessitated robust benchmarks for evaluating LLM performance and adaptability as they incorporated new knowledge and capabilities [28].

NLP's evolution includes specialized methods like SMP-BERT for structured data extraction in fields like medicine, addressing data sparsity and class imbalances challenges [29]. Traditional fine-tuning approaches revealed limitations, especially with limited data, prompting exploration of alternative strategies not reliant on large datasets [30]. The evolution of prompting strategies highlighted task-specific methods' limitations, leading to sophisticated techniques like Self-Guiding Exploration (SGE) to enhance LLM generalizability [13].

Benchmarks like LLM-CODE-EXPL are crucial for assessing LLMs' effectiveness in educational applications, illustrating their potential to transform learning environments [31]. However, existing benchmarks often suffer from inadequate testing and imprecise problem descriptions, resulting in insufficient LLM performance evaluations [25]. This inadequacy necessitates a reassessment of evaluation frameworks, particularly in software engineering, where task complexity is inadequately represented [32]. Generating high-quality synthetic data from LLMs remains a core issue, especially in specialized domains with prevalent data scarcity [33].

Integrating LLMs into traditional database management systems through innovative architectures like DB-first has enabled hybrid querying capabilities, showcasing LLMs' versatility in managing structured data [2]. Despite these advancements, adapting explanations to different user contexts remains inadequately assessed, limiting understanding of LLMs' effectiveness in generating tailored explanations [24].

The evolution of LLMs and NLP reflects a dynamic interplay of innovation and challenge, necessitating ongoing research to address existing models' limitations and fully harness their potential across diverse applications. As the field evolves, developing comprehensive evaluation frameworks is essential for ensuring LLMs' reliability and effectiveness in managing complex language tasks [34]. A survey of LLMs in computing education introduces a framework for categorizing existing research, focusing on their performance, pedagogical applications, and ethical implications [5].

3 Prompt Programming as a New AI Paradigm

3.1 Prompt Programming in AI Paradigms

Prompt programming revolutionizes AI paradigms by enhancing the adaptability and efficiency of large language models (LLMs) across various domains, enabling users to leverage LLM capabilities more effectively [15]. Traditional methods, such as those used with GPT-3, often face challenges in task control and evaluation [9]. Innovations like OpenPrompt offer modular toolkits to facilitate prompt-learning implementation [1]. Strategic applications like Tree Prompting classify data efficiently without altering model parameters, optimizing performance while maintaining model integrity [35]. Prompt compression techniques further enhance language model efficiency, showcasing a shift in AI paradigms [36]. Carefully selected prompt examples improve machine translation capabilities, demonstrating potential for enhanced task performance [11].

In-context learning distinguishes between task representation (TR) and task learning (TL), refining model learning and adaptation processes [37]. This is crucial for utilizing LLMs in processing natural language path instructions, as shown in language-to-topological map conversions [38]. Prompt programming addresses cognitive biases in LLMs through innovative hashing methods, enhancing objectivity and reliability [39]. Techniques like TCP mitigate hallucinations in LLMs, moving towards more reliable AI systems [40].

Integrating natural language reasoning with programming constructs, as seen in PAL, allows LLMs to generate both descriptive and executable code, marking a significant evolution in AI applications for programming and software development [41]. This capability extends to anomaly detection, where

LLMs utilize supervised fine-tuning and in-context learning, showcasing their versatility in analytical tasks [42]. ChartGPT enhances visualization generation from natural language inputs, exemplifying the shift towards more intuitive AI interfaces [43]. Controlled text generation techniques, such as ScoPE, employ progressive editing mechanisms for better management of generated attributes, reflecting a sophisticated approach to AI output control [44].

These advancements in prompt programming actively reshape AI paradigms, driving innovation and expanding LLM capabilities across sectors. Frameworks like AutoFlow, inspired by Automated Machine Learning (AutoML), aim to automate workflow generation, minimizing human effort and optimizing task performance [45]. AutoCompanion integrates LLMs with goal-directed reasoning systems, enhancing conversation management and reflecting a paradigm shift [46]. Developing benchmarks for situationally tailored explanations, as seen in psychology and education, ensures contextually relevant AI outputs [24]. EvalPlus provides a comprehensive framework for evaluating LLM capabilities in code generation, highlighting the importance of robust evaluation mechanisms [25]. CREATOR enhances tool creation, focusing on generality, reusability, and variety, crucial for advancing AI paradigms [3]. Systematic analyses of causal relationships between prompts and generated code enhance LLM programming capabilities, representing a new AI paradigm [23]. IBL, which produces models with various structures without assuming inductive bias, enhances accuracy and interpretability, marking a significant evolution in AI methodologies [47]. MaPLe exemplifies a new approach by simultaneously adapting vision and language modalities, enhancing interaction and improving performance on downstream tasks [22].

3.2 Leveraging NLP as Programming

Integrating natural language processing (NLP) into programming paradigms signifies an evolution in utilizing LLMs for complex tasks. This shift emphasizes NLP techniques as programming tools, where crafted natural language prompts direct LLMs to produce precise and contextually relevant outputs. Techniques like according-to prompting enhance response grounding in existing text corpora, ensuring generated content aligns with factual information while integrating safety-driven guidelines to mitigate risks [48, 4, 49]. This approach transforms human language into a functional programming medium.

As illustrated in Figure 2, the integration of NLP as a programming tool highlights key techniques, educational applications, and interactive systems that enhance accessibility and learning through natural language prompts. In educational contexts, the interaction between students and AI systems like ChatGPT is structured into stages emphasizing feedback loops and new learning behaviors, showcasing the potential of NLP-driven programming to foster adaptive learning environments. LLMs provide tailored feedback, supporting the development of independent learning strategies [10]. By utilizing NLP as a programming tool, educators can design prompts that engage students in dynamic learning processes, enhancing material understanding and retention.

The adaptability of NLP techniques in prompt programming is further illustrated by interactive systems relying on natural language inputs for tasks typically reserved for explicit programming languages. The Prompt Middleware framework enables users to engage with LLMs through conversational prompts, streamlining task execution and enhancing accessibility for non-expert users. This framework facilitates prompt generation based on user interface affordances, including static, template-based, and free-form prompts for tailored interactions. Additionally, advancements in personalized exploratory systems, such as the Collaborative Assistant for Personalized Exploration (CARE), demonstrate how multi-agent LLM frameworks refine user queries and deliver customized solutions, transforming LLMs into proactive collaborators in complex tasks [24, 50, 51, 17]. Leveraging NLP as a programming tool democratizes access to advanced AI capabilities, enabling broader applications across diverse domains.

3.3 Comparison with Traditional Programming

Prompt programming and traditional programming represent distinct paradigms within AI and software development, each characterized by unique methodologies. Traditional programming involves explicit coding using formal languages like Python, Java, or C++, where developers create algorithms to execute tasks. In contrast, prompt programming, exemplified by models like GPT-4, involves crafting prompts to interact with AI models, enabling them to perform tasks based on these

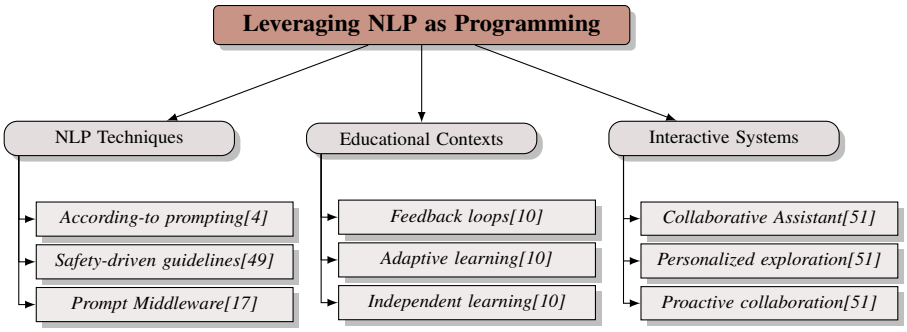


Figure 2: This figure illustrates the integration of NLP as a programming tool, highlighting key techniques, educational applications, and interactive systems that enhance accessibility and learning through natural language prompts.

inputs. This paradigm shift requires developers to cultivate mental models of AI behavior and adapt prompts through iterative experimentation, highlighting the significance of prompt engineering in developing intelligent application features [10, 15].

Prompt programming leverages NLP techniques to guide LLMs in generating desired outputs through crafted prompts, allowing users to interact with AI systems using natural language and reducing the need for extensive programming knowledge. This democratization of AI capabilities enables non-expert users to harness LLM power across various applications [15]. A primary distinction lies in the level of abstraction involved. Traditional programming demands precise, detailed instructions, while prompt programming abstracts task execution complexity by relying on the LLM's ability to interpret and respond to natural language inputs. This abstraction allows greater flexibility, enabling users to modify prompts for varying outcomes without altering underlying code [9].

Prompt programming emphasizes task specification over example-based learning, allowing LLMs to generalize across a broader range of tasks, beneficial in scenarios where explicit programming is cumbersome, such as generating creative content or executing complex reasoning tasks [23]. The ability of LLMs to generate intermediate reasoning steps and execute them through external interpreters further enhances problem-solving capabilities, setting prompt programming apart from traditional methods [22].

Despite its advantages, prompt programming presents challenges related to prompt sensitivity and variability, where minor changes can lead to significant differences in outputs. This sensitivity necessitates robust prompting strategies to ensure consistency and reliability in AI interactions [8]. Additionally, reliance on pre-trained models poses limitations in customization and control over specific functionalities, which traditional programming can more readily address through direct code manipulation.

The comparison between traditional programming and prompt programming underscores the evolution of AI paradigms towards more intuitive and accessible interfaces. As prompt programming advances, it presents unique opportunities for innovation and exploration, expanding AI technology applications across diverse domains [6].

4 Techniques and Strategies in Prompt Programming

The exploration of techniques and strategies in prompt programming unveils a rapidly evolving field that meets the intricate demands of AI and NLP. Innovations in this domain significantly enhance LLM performance and extend their applicability across diverse sectors. As illustrated in Figure 3, the hierarchical structure of these techniques and strategies showcases a variety of innovations and methodologies, including advanced techniques, optimization and stability strategies, automated and iterative prompting strategies, as well as prompt engineering aimed at enhancing model capabilities. This figure categorically outlines key methodologies, applications, and examples of frameworks that collectively enhance the capabilities of large language models (LLMs) across various domains. Table 2 presents a detailed categorization of recent advancements and methodologies in prompt programming, emphasizing their transformative potential in optimizing the capabilities of large

Category	Feature	Method
Innovations and Methodologies in Prompt Programming	Contextual Language Models Adaptive Interaction Tools	IBL[47] CREATOR[3]
Advanced Prompt Programming Techniques	Quality Enhancement	APE[52]
Optimization and Stability in Prompt Programming	Feedback and Refinement Error Detection and Safety Efficiency and Optimization Data Extraction and Utilization	MaPLe[22], VPP[7], PRP[53] OS[54], AIS-LLM[55] DSPy[56], PEZ[20] MZSL[57]
Automated and Iterative Prompting Strategies	Automated Prompt Techniques	PP[9]
Prompt Engineering for Enhanced Model Capabilities	Prompt Optimization	CACGE[23], RM[58], SMP-BERT[29], LLMs-AD[42]
Hybrid and Modular Prompting Frameworks	Integrated Prompting Strategies	AP[16], OP[1], PW[59], PM[17], PLID[21]

Table 1: This table provides a comprehensive overview of recent advancements and methodologies in prompt programming, categorizing them into distinct areas such as innovations and methodologies, advanced techniques, optimization and stability, automated strategies, prompt engineering, and hybrid frameworks. Each category highlights specific features and associated methods, showcasing the diverse strategies employed to enhance the capabilities of large language models (LLMs) across various applications. The references cited offer further insights into the specific contributions and innovations within each category.

Category	Feature	Method
Innovations and Methodologies in Prompt Programming	Contextual Language Models Adaptive Interaction Tools	IBL[47] CREATOR[3]
Advanced Prompt Programming Techniques	Quality Enhancement	APE[52]
Optimization and Stability in Prompt Programming	Feedback and Refinement Error Detection and Safety Efficiency and Optimization Data Extraction and Utilization	MaPLe[22], VPP[7], PRP[53] OS[54], AIS-LLM[55] DSPy[56], PEZ[20] MZSL[57]
Automated and Iterative Prompting Strategies	Automated Prompt Techniques	PP[9]
Prompt Engineering for Enhanced Model Capabilities	Prompt Optimization	CACGE[23], RM[58], SMP-BERT[29], LLMs-AD[42]
Hybrid and Modular Prompting Frameworks	Integrated Prompting Strategies	AP[16], OP[1], PW[59], PM[17], PLID[21]

Table 2: This table provides a comprehensive overview of recent advancements and methodologies in prompt programming, categorizing them into distinct areas such as innovations and methodologies, advanced techniques, optimization and stability, automated strategies, prompt engineering, and hybrid frameworks. Each category highlights specific features and associated methods, showcasing the diverse strategies employed to enhance the capabilities of large language models (LLMs) across various applications. The references cited offer further insights into the specific contributions and innovations within each category.

language models (LLMs). Table 7 offers a comprehensive comparison of recent advancements and methodologies in prompt programming, emphasizing their transformative potential in optimizing the capabilities of large language models (LLMs). The following subsection will highlight recent advancements and methodologies in prompt programming, emphasizing their transformative potential in optimizing LLM capabilities.

4.1 Innovations and Methodologies in Prompt Programming

Method Name	Methodological Advancements	Contextual Interaction	Application Impact
PLID[21]	Plid Method	Informative Class Distributions	Enhanced Compositional Recognition
MaPLe[22]	Joint Prompting Approach	Contextual Prompts Creation	Transformative Potential Education
CREATOR[3]	Tool Creation Framework	Contextually Relevant Prompts	Transformative Potential Education
IBL[47]	Inductive-Bias Learning	Contextual Understanding Llms	Practical Applications Inference

Table 3: Overview of recent methodological advancements in prompt programming, highlighting the innovative frameworks and their impact on contextual interaction and application domains. The table delineates the contributions of various methods, including PLID, MaPLe, CREATOR, and IBL, to the enhancement of compositional recognition, educational transformation, and practical inference applications.

Recent advancements in prompt programming have introduced methodologies that significantly boost LLM capabilities. Table 3 provides a comprehensive summary of recent innovations in prompt programming methodologies, illustrating their contributions to enhancing LLM capabilities through contextually informed interactions and transformative applications. Language-informed

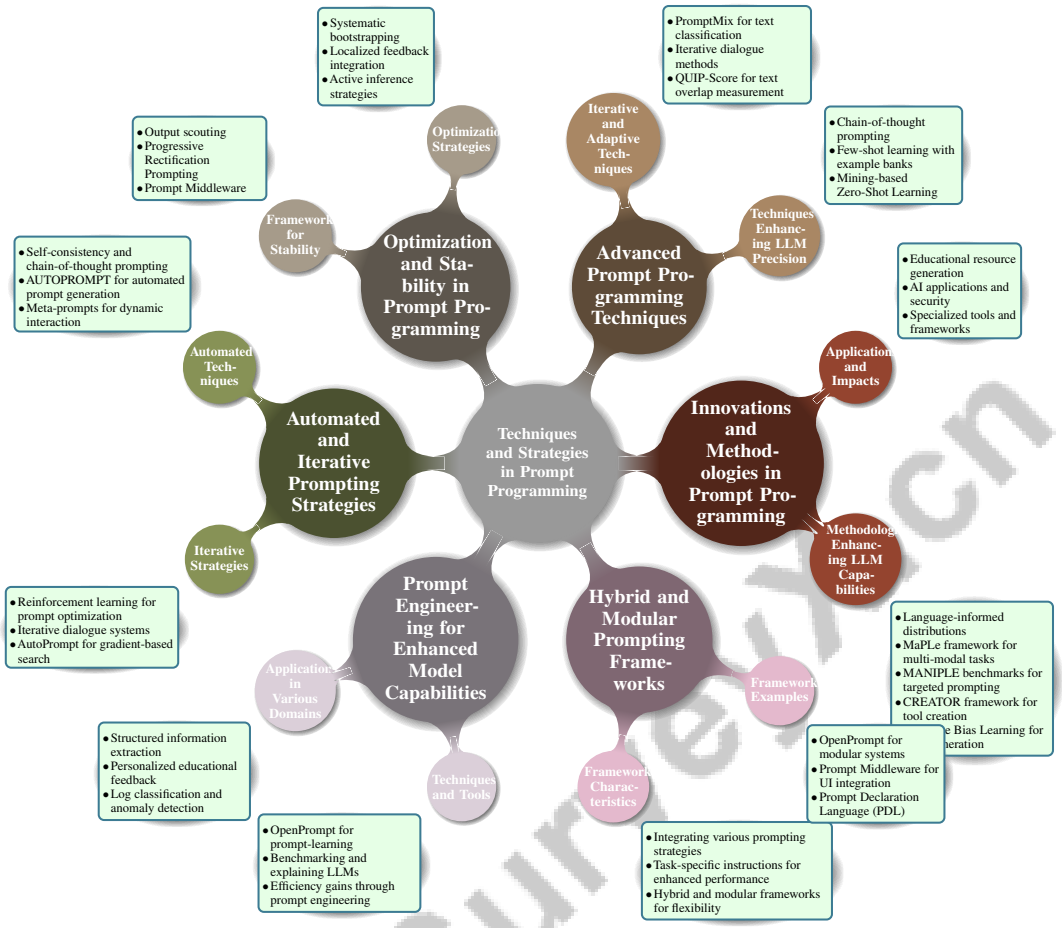


Figure 3: This figure illustrates the hierarchical structure of techniques and strategies in prompt programming, showcasing innovations and methodologies, advanced techniques, optimization and stability strategies, automated and iterative prompting strategies, prompt engineering for enhanced model capabilities, and hybrid and modular prompting frameworks. Each category outlines key methodologies, applications, and examples of frameworks that enhance the capabilities of large language models (LLMs) across various domains.

distributions enable nuanced interactions by creating contextually relevant prompts, contrasting with deterministic methods [21]. The MaPLe framework exemplifies advancements by jointly learning contextual prompts in vision and language branches, enhancing multi-modal task performance [22]. Benchmarks like MANIPLE highlight the importance of targeted prompting strategies, showing that customized fact selection enhances LLM performance in specific applications [26]. The CREATOR framework fosters dynamic interactions between LLMs and task environments by enabling tool creation through a structured process [3]. Inductive Bias Learning (IBL) for Python code generation demonstrates how LLMs' contextual understanding can enhance code generation [47].

In educational contexts, LLMs reduce instructor workload through automated resource generation, illustrating their transformative potential in pedagogy [5]. These innovations signify a transformative advancement in AI and NLP, particularly evident in generative pre-trained models like GPT-4, which catalyze prompt engineering—an iterative process optimizing AI performance. These methodologies enhance AI applications and address AI security and model robustness against adversarial attacks. Insights from prompt programming studies reveal its distinct nature from traditional software development, highlighting the need for specialized tools and frameworks. Collectively, these advancements expand AI and NLP technologies' potential applications, impacting millions through intelligent software solutions [15, 60].

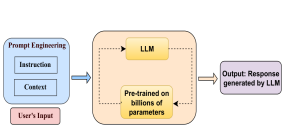
4.2 Advanced Prompt Programming Techniques

Advanced prompt programming techniques have significantly bolstered LLM capabilities, enabling them to tackle complex tasks with precision. Chain-of-thought (CoT) prompting structures reasoning steps logically, improving performance on structured reasoning tasks [21]. Few-shot learning techniques, like utilizing example banks of buggy-fixed pairs, enhance multilingual capabilities crucial for automated program repair [26]. Mining-based Zero-Shot Learning (MZSL) leverages labeled examples mined from an unlabeled corpus using regex patterns and verbalizers, improving accuracy across reasoning benchmarks.

PromptMix, a two-step prompting method, enhances text classification by generating and relabeling borderline examples, demonstrating its dual-step process mitigates false positives and effectively transfers knowledge from large models to smaller classifiers like DistilBERTbase and BERTbase [61, 62, 16, 63, 20]. Iterative dialogue methods illustrate adaptive characteristics, enabling LLMs to decompose complex problems into manageable subproblems through recursive questioning, integrating solutions into cohesive responses [64, 65, 66, 67, 50].

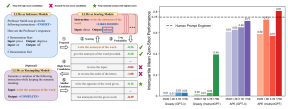
The investigation into the relationship between reasoning step length and model accuracy reveals a significant linear correlation, with optimal performance at six reasoning steps, underscoring the critical role of structured, stepwise reasoning in enhancing LLM capabilities during complex problem-solving tasks [59, 68, 69]. Techniques such as direct task specification, task specification by demonstration, and metaprompt programming refine prompts, showcasing diverse strategies in advanced prompt programming.

Collectively, these advanced techniques underscore prompt programming's transformative potential in enhancing LLM capabilities, driving innovation across a wide range of AI and NLP applications. As the field evolves, exploring and refining these methodologies will be essential for unlocking LLMs' full potential in tackling increasingly complex tasks. The QUIP-Score metric efficiently measures the overlap between generated text and pre-training data, distinguishing it from previous benchmarks lacking precise quantification [4].



(a) Prompt Engineering Process Flowchart[62]

(b) Code Prompt Examples with Different Variants[18]



(c) LLMs as Inference Models: A Comparative Study on Prompt Selection for Semantic Reasoning Tasks[52]

Figure 4: Examples of Advanced Prompt Programming Techniques

As shown in Figure 4, advanced prompt programming techniques employ various strategies and methodologies to enhance language models' effectiveness. The accompanying figure showcases three pivotal examples illustrating these techniques. Firstly, the "Prompt Engineering Process Flowchart" visually represents the systematic approach to crafting prompts for language models, detailing the journey from user input to response generation via pre-trained models. Secondly, "Code Prompt Examples with Different Variants" delves into the nuances of code prompt construction, presenting a comparative analysis of four distinct formats, each with unique identifiers and comments to optimize task execution. Lastly, "LLMs as Inference Models: A Comparative Study on Prompt Selection for Semantic Reasoning Tasks" explores various language models' comparative performance in selecting the most suitable prompts for complex reasoning tasks. Together, these examples underscore the sophistication and versatility of prompt programming techniques, offering insights into strategically manipulating prompts to achieve desired outputs from language models [62, 18, 52].

4.3 Optimization and Stability in Prompt Programming

Optimization and stability in prompt programming are critical for enhancing the performance and reliability of LLMs across diverse applications. Table 4 provides a comprehensive comparison of different prompt programming methods, detailing their optimization techniques, reliability mechanisms, and application scenarios, thereby illustrating the varied approaches to enhancing large language mod-

Method Name	Optimization Techniques	Reliability Mechanisms	Application Scenarios
DSPy[56]	Systematic Bootstrapping	Learned Demonstrations	Software Applications
VPP[7]	Automated Verification	Localized Feedback	Network Configurations
AIS-LLM[55]	Actor-critic Prompting	Active Inference	Medical Applications
OS[54]	Output Scouting	Localized Feedback Integration	User Interfaces
MZSL[57]	Enhanced Filtering Techniques	Interactive Design Patterns	Text Classification Tasks
PRP[53]	Systematic Bootstrapping	Output Scouting	User Interfaces
PEZ[20]	Gradient-based Methods	Statistical Reliability	Generative Models
MaPLe[22]	Prompt Learning Process	Better Robustness Evaluations	Vision-language Tasks

Table 4: Overview of optimization techniques, reliability mechanisms, and application scenarios for various prompt programming methods in large language models (LLMs). This table highlights the diverse strategies employed across different methods to enhance performance, reliability, and applicability in software, network, medical, and user interface contexts.

els (LLMs) across multiple domains. Systematic bootstrapping of demonstrations and optimizations, as seen in DSPy, enhances performance without hand-crafted prompts [56]. Integrating localized feedback from verification tools corrects configuration errors and improves LLM output accuracy [7]. Active inference strategies, incorporating validated information into LLM responses, exemplify robust frameworks ensuring AI output reliability and safety [55]. Output scouting systematically identifies catastrophic responses in LLM outputs, maintaining AI-generated content's integrity and trustworthiness [54].

Mining-based approaches outperform traditional prompting in zero-shot learning tasks, demonstrating improved performance and flexibility [57]. Progressive Rectification Prompting (PRP) addresses iterative refinement challenges in complex tasks, underscoring the need for balancing optimization efficiency with computational resources [53]. PEZ offers a streamlined approach to generating high-quality prompts using fewer tokens, maintaining interpretability and usability across different models [20]. Despite the benefits of learnable prompts, increased complexity may not suit all applications [22].

Integrating LLMs into user interfaces and software applications necessitates optimizing and stabilizing prompt programming processes for effective deployment across various domains. Frameworks like Prompt Middleware facilitate this by generating prompts tailored to UI affordances. As research progresses, developing more sophisticated strategies will be crucial for maximizing LLMs' potential in delivering accurate and reliable AI solutions [15, 17].

4.4 Automated and Iterative Prompting Strategies

Method Name	Automation Techniques	Iterative Refinement	Model Adaptability
PP[9]	Automated Prompt Generation	Metaprompt Programming	Advanced Prompting Strategies
OP[1]	Automatic Verbalizers	Continuous Improvement Cycles	Advanced Prompting Strategies
PLID[21]	Llm-generated Descriptions	Continuous Improvement Cycles	Dynamic Class Embeddings

Table 5: Comparison of automated and iterative prompting strategies across different methods, highlighting the use of automation techniques, iterative refinement processes, and model adaptability. The table illustrates the diverse approaches employed by various methodologies, such as prompt generation, metaprompt programming, and dynamic class embeddings, in enhancing the performance and adaptability of large language models (LLMs).

Automated and iterative prompting strategies have emerged as essential techniques for optimizing the efficiency and effectiveness of prompt programming tasks, particularly for LLMs like GPT-4 and Claude-3. Advanced methodologies such as self-consistency and chain-of-thought prompting enable users to structure inputs more effectively, maximizing models' utility and accuracy. The AUTOPROMPT method demonstrates automated prompt generation's potential to significantly enhance model performance across diverse tasks, achieving results comparable to state-of-the-art supervised models without additional parameters or fine-tuning [16, 60]. Automation and iterative refinement streamline the crafting process, reducing manual effort and improving AI output consistency. Table 5 presents a comprehensive comparison of automated and iterative prompting strategies, demonstrating their impact on optimizing prompt programming tasks for large language models.

Meta-prompts enable LLMs to autonomously generate their own prompts based on predefined criteria, facilitating dynamic interaction with the environment and continuous prompt adaptation [9].

Iterative prompting strategies often employ reinforcement learning to optimize prompt generation, enhancing task performance by allowing iterative refinement based on feedback [59, 70, 63, 20]. Automated frameworks like AutoPrompt exemplify automation's potential in prompt programming, using gradient-based search methods to discover prompts that maximize model performance [1].

Developing iterative dialogue systems allows LLMs to engage in recursive interactions, refining responses through question-and-answer exchanges. This iterative process is valuable in complex problem-solving scenarios, where a single prompt may not suffice to capture the full task scope [21]. Automated and iterative prompting strategies enhance LLM adaptability and effectiveness, paving the way for more robust and versatile AI applications. As LLMs advance, developing innovative methodologies for automating and refining prompt programming tasks will be essential [48, 71, 72].

4.5 Prompt Engineering for Enhanced Model Capabilities

Method Name	Application Domains	Techniques and Tools	Outcome Improvements
SMP-BERT[29]	Radiology Reports	Section Matching Prediction	Higher F1-scores
LLMs-AD[42]	Anomaly Detection	Supervised Fine-tuning	Improved Performance
OP[1]	Text Classification	Modular Toolkit	Enhanced Performance
CACGE[23]	Code Generation	Rephrasing Techniques	Code Generation Quality
RM[58]	Reasoning Tasks	Reinforcement Learning	Reduction IN Costs

Table 6: Overview of various prompt engineering methods and their application domains, techniques, and outcome improvements. This table highlights the diverse methodologies employed to enhance the capabilities of large language models across different tasks, demonstrating the tangible benefits in performance and cost efficiency.

Prompt engineering has emerged as a pivotal technique for enhancing LLM capabilities by optimizing how these models interpret and respond to input prompts. Strategically designing prompts allows researchers to significantly improve model output accuracy and reliability across various tasks. A notable application of prompt engineering is in structured information extraction, where techniques like those used in SMP-BERT effectively handle structured sections of radiology reports, enhancing the model's capabilities in specialized fields [29]. Table 6 provides a comprehensive summary of prompt engineering methods, illustrating their application in enhancing model capabilities across multiple domains.

In educational contexts, prompt engineering plays a critical role in personalizing and enhancing learning experiences for novice programmers. The potential of LLMs to provide detailed feedback on programming exercises is demonstrated by benchmarks focusing on generating personalized feedback, thus improving educational outcomes for learners [73]. This aligns with the broader trend of using LLMs to classify log entries as normal or anomalous, leveraging learned patterns from labeled data to enhance anomaly detection capabilities [42].

Integrating prompt engineering techniques is further exemplified by developing datasets that include labeled dialogue messages, such as those annotated with Big Five personality traits. These datasets underscore the importance of prompt engineering in generating high-quality, labeled training data for various NLP tasks [14]. Structured approaches like OpenPrompt facilitate prompt-learning by integrating pre-trained language models (PLMs), templates, and verbalizers, streamlining the prompt engineering process and enhancing model capabilities [1].

Insights from studies on benchmarking and explaining LLMs provide valuable guidance for developers in crafting more effective prompts, emphasizing optimized prompt design in enhancing model capabilities [23]. Recent advancements demonstrate significant reductions in inference costs while maintaining comparable task performance across various datasets, showcasing efficiency gains achieved through effective prompt engineering [58]. Collectively, these developments underscore the transformative potential of prompt engineering in expanding AI technologies' applications across diverse domains.

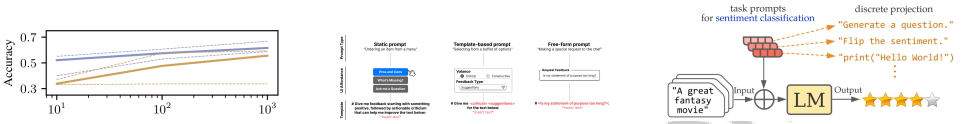
4.6 Hybrid and Modular Prompting Frameworks

Hybrid and modular prompting frameworks signify a major advancement in prompt programming, providing a versatile and scalable methodology for optimizing LLM functionality. These frameworks integrate various prompting strategies—such as predefined, template-based, and free-form

prompts—enabling developers to tailor LLM interactions effectively for diverse applications. By leveraging task-specific instructions, these frameworks enhance model performance without altering underlying model parameters, streamlining LLM deployment in user interfaces and other practical scenarios [62, 1, 17, 60, 71].

Hybrid frameworks incorporate multiple prompting strategies, such as chain-of-thought prompting and few-shot learning, into a cohesive system, enhancing LLMs' ability to tackle complex tasks requiring structured reasoning and contextual understanding [21]. Modular frameworks allow independent development and integration of different prompt components, facilitating customization for specific applications without altering the entire system. The development of frameworks like OpenPrompt exemplifies the potential of modular prompting systems [1].

Hybrid and modular frameworks also address traditional prompting methods' limitations, ensuring more consistent and reliable AI outputs by combining different strategies. The adoption of these frameworks marks a transformative leap in prompt programming, providing flexible and scalable methodologies for enhancing LLM functionality. Frameworks like Prompt Middleware and the Prompt Declaration Language (PDL) facilitate AI integration into user interfaces and streamline prompt creation, ultimately unlocking LLM capabilities' full potential [60, 71, 17].



(a) Accuracy[16]

(b) Prompt Types and Feedback UI Affordances in a Text Generation Interface[17]

(c) A diagram illustrating a machine learning model for sentiment classification[59]

Figure 5: Examples of Hybrid and Modular Prompting Frameworks

As shown in Figure 5, exploring hybrid and modular prompting frameworks offers a nuanced approach to enhancing machine learning models' efficacy and adaptability. The example discussed here, illustrated through a series of images, delves into various techniques and strategies underpinning this innovative framework. The first image focuses on accuracy, depicting a line graph comparing datasets across a logarithmic scale to highlight performance metrics, underscoring dataset selection's significance in optimizing model accuracy. The second image shifts attention to the user interface aspect, showcasing different prompt types and feedback mechanisms within a text generation interface, categorizing prompts into static, template-based, and free-form types, while detailing how feedback UI affordances facilitate user interaction and feedback. Lastly, the third image presents a diagram of a machine learning model designed for sentiment classification, illustrating the process from input to output and emphasizing task-specific prompts' role in refining model predictions. Together, these examples encapsulate the diverse strategies employed in hybrid and modular prompting frameworks, demonstrating their potential to drive advancements in machine learning applications [16, 17, 59].

Feature	Innovations and Methodologies in Prompt Programming	Advanced Prompt Programming Techniques	Optimization and Stability in Prompt Programming
Optimization Technique	Contextual Prompts	Chain-of-thought	Systematic Bootstrapping
Application Scenario	Multi-modal Tasks	Complex Reasoning	Error Correction
Model Performance	Enhanced Capabilities	Improved Accuracy	Increased Reliability

Table 7: This table provides a comparative analysis of various prompt programming methodologies, highlighting the optimization techniques, application scenarios, and model performance outcomes associated with each approach. It categorizes the innovations and methodologies in prompt programming into three key areas: Innovations and Methodologies, Advanced Techniques, and Optimization and Stability, illustrating their impact on enhancing the capabilities of large language models (LLMs). The table underscores the transformative potential of these strategies in optimizing LLM performance across diverse applications.

5 Applications of Prompt Programming in LLMs

The integration of prompt programming within large language models (LLMs) has catalyzed innovation across numerous fields. This section explores its diverse applications, particularly in specialized

domains such as robotics, healthcare, data analysis, and education, showcasing how LLMs tackle complex challenges and enhance operational efficiencies. Subsequent subsections will delve into the role of prompt programming in these domains, highlighting its impact on improving functionalities and outcomes.

5.1 Prompt Programming in Specialized Domains

Prompt programming has revolutionized various specialized domains by enhancing LLM capabilities to address complex, domain-specific challenges. In robotics, it advances navigation and tool use, with frameworks like Tree of Thoughts improving problem-solving in planning and exploration tasks, and RoboTool expanding robotic functionalities [74]. In healthcare, LLMs streamline communication, documentation, and patient education, enhancing care delivery [75]. Data analysis benefits from template-based prompts, as demonstrated by FeedbackBuffet, which automates feedback generation, improving data analysis quality [17].

Robotic systems' robustness against prompt injection attacks has been evaluated to improve security and reliability in navigation [76]. In education, techniques like Progressive Rectification Prompting (PRP) have improved math problem-solving accuracy significantly [53]. The MaPLe framework enhances generalization in novel classes and cross-dataset evaluations, showcasing its application in specialized domains [22]. LLM-assisted programming reveals the impact of reflection on artistic performance, emphasizing prompt programming's role in creative domains [77]. Auto-CoT's effectiveness in reasoning tasks further demonstrates prompt programming's role in cognitive and computational tasks [68].

These applications highlight prompt programming's transformative potential in specialized domains, driving innovation and expanding LLMs' functional scope across diverse fields. As AI advances, refining prompt programming techniques is crucial for fully leveraging AI technologies to tackle complex, domain-specific challenges. Generative pre-trained models like GPT-4 have spurred the development of prompt engineering, where users iteratively create and refine prompts to optimize performance. This process differs from traditional software development, necessitating tailored tools and methodologies to support this unique practice. Effective prompt engineering enhances LLM utility and addresses critical concerns such as AI security and adversarial vulnerabilities, paving the way for robust AI applications across domains [15, 60].

5.2 Robotics and Navigation

Incorporating prompt engineering into robotics and navigation systems has significantly enhanced LLM performance and reliability, enabling them to process multi-modal inputs and generate contextually aware responses while addressing security vulnerabilities and adversarial attacks [62, 17, 76, 60, 71]. LLMs improve complex navigation tasks' efficiency and precision, particularly in embodied AI, where natural language prompts enhance human-robot interaction.

However, integrating LLMs into robotics raises security concerns, especially regarding prompt injection attacks that can mislead navigation, posing substantial threats to system safety and reliability [76]. Addressing these vulnerabilities is crucial for secure LLM-enhanced robotics deployment.

Prompt programming also aids in summarizing scientific concepts and streamlining data extraction in medical research, demonstrating LLMs' versatility in handling complex information tasks [75]. Efficiently summarizing and extracting data from research publications enhances LLM functionality in medical applications, advancing healthcare technologies.

The integration of prompt programming in robotics and navigation illustrates LLMs' transformative potential in enhancing AI capabilities across domains, as evidenced by advancements in multi-modal prompt processing that improve context-aware responses while highlighting the importance of addressing security challenges, such as prompt injection attacks, to ensure safe and effective robot operations [76, 60, 17]. As the field evolves, addressing security challenges and optimizing prompt strategies will be essential for maximizing LLMs' benefits in robotics and navigation.

5.3 Data Analysis and Industry Impact

Prompt programming profoundly impacts data analysis and various industries, significantly enhancing LLM capabilities to execute complex tasks with improved accuracy and efficiency. In data analysis,

LLMs demonstrate remarkable proficiency in handling structured tasks, underscoring their potential in complex data environments [78]. Integrating LLMs with traditional database systems, as exemplified by Galois, advances data retrieval processes, enabling flexible and efficient access to diverse data sources [2].

In industry applications, LLMs enhance workflow automation and efficiency. The AutoFlow framework exemplifies this by improving workflow performance compared to manual designs, reducing human labor and enhancing automation [45]. This capability is crucial in sectors requiring precise and efficient data processing, such as e-commerce and market research, where LLMs surpass traditional models in classification accuracy [79].

Innovative tools like ChartGPT highlight LLMs' transformative impact on data visualization, outperforming existing methods in generating accurate charts from natural language inputs [43]. Scalable evaluation methods for chatbots provide insights into performance without extensive human involvement [80].

In educational settings, LLM-generated tips for complex subjects like quantum computing enhance educational tools [81]. Prompt programming's impact on specialized applications, such as lie detection in dialogues, further illustrates LLMs' potential to improve niche areas' performance, offering enhanced accuracy over existing methods [82].

Prompt programming facilitates LLM integration into various industries, driving innovation and improving data analysis processes. As research progresses, advanced prompt programming strategies will be crucial for enhancing LLM effectiveness in data analysis and industrial applications. This development is essential for maintaining LLM relevance in tackling complex challenges, enabling developers to create tailored prompts that optimize performance and user satisfaction. Understanding prompt programming nuances—such as iterative prompt refinement and robust mental models of model behavior—allows developers to harness LLM capabilities effectively, ensuring these technologies continue providing valuable insights and solutions across domains [19, 71, 15, 50].

5.4 Image and Text Classification

Prompt programming has significantly advanced LLM capabilities in image and text classification tasks, enabling precise, context-aware classifications across domains. This is evident in multi-modal frameworks leveraging visual and textual data to enhance accuracy. The MaPLe framework exemplifies this by learning prompts in both vision and language branches, improving multi-modal task performance [22].

In text classification, prompt programming facilitates techniques like Mining-based Zero-Shot Learning (MZSL), enhancing zero-shot capabilities by mining labeled examples from unlabeled corpora using regex patterns and verbalizers. This approach simplifies learning and improves accuracy across reasoning benchmarks [57]. Advanced techniques like chain-of-thought prompting improve text classification by structuring reasoning steps logically, facilitating clearer outcomes [21].

In image classification, integrating LLMs with vision-language models enables nuanced interactions with visual data, enhancing classification based on contextual information from natural language prompts. This is beneficial in applications requiring concurrent visual and textual input processing, such as classifying images with descriptive text. Leveraging advanced vision-language models like CLIP aligns visual and textual data in a shared feature space, improving tasks like image recognition and sentiment analysis. Techniques like multi-modal prompt learning and prompt-aware adapters enhance visual and textual information integration, enabling models to focus on relevant features and improve interpretability [70, 83, 57, 22, 84].

Overall, prompt programming in image and text classification underscores LLMs' transformative potential in enhancing AI capabilities across domains. By leveraging multi-modal frameworks and advanced prompting techniques, prompt programming drives innovation in classification tasks, offering accurate, context-aware solutions in both image and text domains. As LLMs advance, exploring and refining prompt programming strategies will be crucial for maximizing their effectiveness in tackling intricate classification challenges. Recent research shows LLM response quality improves with carefully crafted prompts, as demonstrated by ExpertPrompting, which tailors prompts for expert-level answers. Innovative approaches like Prompt Problems are reshaping programming education by enabling students to translate problem statements into effective prompts, enhancing

LLM interaction. These developments underscore sophisticated prompting techniques' importance in harnessing LLMs' full capabilities for complex tasks [19, 50].

5.5 Education and Instructional Design

Prompt programming plays a transformative role in educational and instructional design, enabling innovative tools and methodologies that enhance learning experiences. LLMs like ChatGLM generate high school IT exam questions, impacting educational content design and delivery [66]. This capability allows educators to create customized assessments aligned with learning objectives, fostering a tailored educational environment.

In medical education, LLMs support learning through customized educational tools and simulations, offering students dynamic engagement with complex medical concepts [75]. Prompt programming enables educators to design simulations mimicking real-world scenarios, enhancing critical thinking and decision-making skills.

Prompt Problems have emerged as effective pedagogical tools for teaching programming concepts and fostering metacognitive skills [19]. Strategic prompts guide students in exploring programming challenges, encouraging reflection on problem-solving strategies and deeper programming understanding.

Integrating prompt programming into educational and instructional design represents significant advancement, offering educators tools for more engaging learning experiences. As LLM integration in education progresses, investigating and enhancing prompt programming techniques is crucial. This approach facilitates effective LLM application for teaching and learning, aligning with emerging pedagogical methods, such as using visual problem representations to guide students in crafting prompts for accurate code generation. Refining these techniques allows educators to harness LLM capabilities, improving educational outcomes across contexts and addressing prompt programming challenges, which differ from traditional software development practices [19, 15].

5.6 Healthcare and Medical Applications

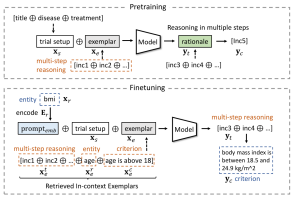
Prompt programming has significantly advanced LLM applications in healthcare and medical fields, offering innovative solutions that enhance medical processes' efficiency and accuracy. AutoTrial demonstrates superior performance in generating clinical trial eligibility criteria, reducing trial failure risks and underscoring prompt programming's transformative impact on clinical research [85].

Structured actor-critic approaches in prompt programming enhance LLM response accuracy and reliability, making them suitable for medical applications. This method ensures consistent, dependable outputs, crucial in high-stakes environments like healthcare, where precision is paramount [55].

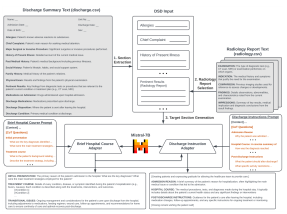
Prompt programming in healthcare extends to improving patient education and communication, where LLMs generate clear content aiding patient understanding of medical conditions and treatments. This underscores LLMs' role in improving patient engagement and empowerment by providing personalized, empathetic responses in healthcare settings, contributing to enhanced health outcomes. Advanced prompting techniques and domain-specific datasets enable LLMs to deliver reliable medical knowledge and therapeutic support, addressing the need for accessible mental health care amid a professional shortage [55, 86].

Implementing prompt programming in healthcare and medical fields illustrates LLMs' transformative potential in revolutionizing medical practices. Prompt engineering techniques, involving iterative prompt development to optimize performance, enable LLMs to deliver precise, dependable, contextually tailored information. This enhances medical data accuracy and facilitates AI-driven tool integration into user interfaces, allowing healthcare professionals efficient access to expert-level insights. As prompt development evolves, it highlights unique challenges and opportunities in software development, paving the way for more effective, responsive medical applications [15, 17]. Exploring and refining prompt programming techniques will be essential for maximizing LLMs' healthcare impact, ensuring continued relevance and effectiveness in addressing complex medical challenges.

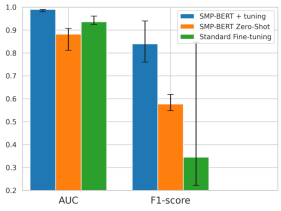
As shown in Figure 6, the integration of large language models (LLMs) into healthcare and medical applications is gaining significant attention. Visual examples illustrate prompt programming's potential in enhancing medical processes. A multi-step reasoning process highlights intricate pre-



(a) The image depicts a multi-step reasoning process in the context of pretraining and fine-tuning for a model.[85]



(b) A flowchart illustrating the process of generating discharge summaries and radiology reports from medical records[87]



(c) The image compares the performance of three different methods in terms of AUC and F1-score.[29]

Figure 6: Examples of Healthcare and Medical Applications

training and fine-tuning stages for generating medical scenario rationales. Another flowchart maps discharge summaries and radiology reports generation from patient records, emphasizing structured patient information handling. A comparative analysis of three methods—SMP-BERT with tuning, SMP-BERT Zero-Shot, and Standard Fine-tuning—demonstrates efficacy in terms of AUC and F1-score, providing insights into performance and reliability in medical contexts. These examples underscore LLMs' transformative potential in healthcare, paving the way for efficient, accurate medical documentation and analysis [85, 87, 29].

5.7 Programming and Software Engineering

Prompt programming significantly influences programming and software engineering by enhancing LLM capabilities to automate and optimize tasks. The CODES system exemplifies this by generating complete code repositories from natural language requirements, streamlining software development [88].

In automated bug replay, prompt programming surpasses traditional methods. Tools evaluated against state-of-the-art methods like ReCDroid and MaCa confirm its effectiveness, highlighting prompt programming's potential in improving software reliability and reducing manual debugging [89].

Prompt programming techniques in model-driven software engineering are facilitated by frameworks like Guidance, using grammar masking for accurate, context-aware outputs [90]. Such methodologies refine software models and improve software engineering practices.

Lessons from AI-driven software solutions like StackSpot AI provide insights into integrating LLMs into development workflows. While based on a single team's experiences, they highlight prompt programming's transformative potential in software engineering [91].

Prompt programming in programming and software engineering exemplifies LLMs' transformative potential in automating and optimizing software tasks. As software engineering integrates LLMs, exploring and refining prompt programming techniques is crucial for enhancing model effectiveness in tackling complex programming challenges. This evolution ensures LLM relevance in software practices and facilitates innovative teaching methods like "Prompt Problems," leveraging LLMs to improve coding comprehension and problem-solving skills. Understanding prompt programming nuances—where developers create and iterate prompts for desired outcomes—is essential for maximizing LLM utility in automating tasks like code generation, requirements elicitation, and software design. Addressing prompt programming's unique challenges and methodologies can significantly improve LLM impact in software engineering [19, 48, 92, 15].

6 Challenges and Limitations

Understanding the challenges and limitations of large language models (LLMs) requires an examination of prompt sensitivity and variability, which significantly impact their performance and reliability. This section explores how prompt design influences model outputs, revealing inherent fragility and variability in LLM interactions. Addressing these issues is crucial for improving prompt engineering and enhancing LLM deployment across various applications.

6.1 Prompt Sensitivity and Variability

Prompt sensitivity and variability pose significant challenges in LLM deployment, often resulting in inconsistent and unpredictable outputs. Minor changes in input prompts can cause substantial variations in model outputs, a phenomenon known as "waywardness," highlighting a disconnect between intended tasks and model performance. This instability necessitates improved prompt engineering methodologies to enhance LLM reliability and mitigate vulnerabilities to adversarial attacks [59, 60].

Designing effective extraction patterns remains a challenge in prompt programming, often leading to suboptimal results. Variability in code quality due to different natural language prompts complicates prompt optimization. Current benchmarks inadequately measure the impact of reasoning step length in chain-of-thought (CoT) prompting, creating a knowledge gap in CoT implementation for improved performance. Studies show that extending reasoning steps in CoT prompts enhances capabilities across datasets, while reduction can severely hinder performance [68, 69]. Existing methods' inability to verify answer correctness and rectify reasoning path errors results in repeated inaccuracies, underscoring the need for better verification mechanisms.

In translation tasks, challenges arise from inconsistent quality and difficulties generalizing optimal demonstration examples across contexts. Research indicates that in-context learning effectiveness in LLMs heavily relies on the quality and selection of demonstration examples, with poor choices degrading performance. Prompting strategies for machine translation underscore the importance of both the number and quality of examples for successful outcomes [93, 4, 49, 11]. The evolving nature of jailbreak techniques poses limitations for current benchmarks, which may not capture all vulnerabilities. Moreover, over-reliance on AI tools in education can lead to superficial learning, necessitating balanced AI integration to foster critical thinking.

The QUIP-Score, focusing solely on exact lexical matches, may overlook other grounding forms, limiting its broader applicability [4]. Addressing prompt sensitivity and variability challenges requires ongoing research to refine prompt design and evaluation methodologies, thereby realizing LLMs' full potential in diverse real-world scenarios.

6.2 Reliability and Trust in AI Outputs

Reliability and trustworthiness of AI outputs from LLMs are critical considerations for their deployment across various domains. A major concern is hallucinations, where models generate fabricated content disconnected from input data. This issue has significant implications as AI platforms gain traction across industries. Recent research highlights methods to mitigate hallucinations, such as contextually tagged prompts, achieving up to 98.88% effectiveness in eliminating inaccuracies. Techniques like "according-to prompting" improve model grounding by directing LLMs to reference prior text, enhancing output accuracy [40, 4]. Robust methodologies are needed to address hallucinations and bolster trust in AI-generated outputs, with proposals like Trapping and Controlling Prompting (TCP) enhancing reliability.

Despite LLMs' potential for accurate results, their opaque generation mechanisms complicate interpretability and trust, challenging users' reliance on AI outputs. Limitations of Inductive Bias Learning (IBL), such as instability in generated code models and accuracy maintenance with larger datasets, further complicate reliable AI output generation [47].

Operational costs associated with techniques like Self-Guiding Exploration (SGE) raise additional concerns about reliability, given the resource intensity of such methods [13]. Achieving exact matches in complex tasks remains challenging, despite superior performance from models like GPT-4o.

Prompt injection attacks exacerbate reliability concerns, particularly in LLM-integrated robotic systems, highlighting the need for robust benchmarks to enhance AI application safety. Current benchmarks often simplify complex real-life scenarios, limiting their effectiveness in diverse contexts [24].

To effectively harness AI across domains, addressing reliability and trustworthiness concerns is crucial, particularly regarding biased content generation and the need for rigorous validation of AI-assisted analyses. Strategies like the Guide-Align approach can enhance output safety and quality by establishing comprehensive guidelines for diverse inputs, ensuring accurate verification of AI-

AI-generated, for reference only.

generated analyses [94, 49]. Continued research into sophisticated methods for enhancing AI output reliability will be essential for maximizing LLM potential in real-world applications.

6.3 Manual Prompt Crafting and Resource Intensity

Manual prompt crafting for LLMs presents notable challenges due to its significant resource intensity, hampering scalability and efficiency in AI deployment. The intricate process of selecting and structuring prompts is burdensome for tasks involving complex reasoning or specialized knowledge, as demonstrated by systems like DialCoT, which require substantial manual effort [95]. This labor-intensive approach is compounded by the need for human intervention in addressing complex errors, as seen in frameworks like VPP, which, while reducing workload, still necessitate manual oversight [7].

Resource intensity is exacerbated by reliance on existing datasets that may not fully capture the diversity of real-world programming challenges, limiting benchmark applicability [25]. The challenges of manual prompt crafting are particularly pronounced in languages with less training data, where frameworks such as RING struggle to generate effective prompts [96]. Additionally, catastrophic failures in zero-shot prompting highlight the limitations of manual crafting, as models may entirely fail to undertake tasks, resulting in misleading performance metrics [9].

The computational cost of prompt crafting is substantial, requiring extensive resources on state-of-the-art hardware [36]. This resource intensity underscores the need for more efficient methodologies, as non-interpretable tokens in hard prompts can lead to unexpected outcomes [20]. Similar challenges in generating quality labeled datasets further emphasize the demands of manual prompt crafting [14].

Despite streamlining efforts from tools like OpenPrompt, users must possess a foundational understanding of prompt-learning concepts to leverage their potential fully [1]. Furthermore, the high cost of using LLMs for prompting and challenges in retrieval efficiency when managing extensive training sets present significant barriers [97].

To enhance the scalability and effectiveness of LLM applications, it is essential to address the challenges posed by manual prompt crafting and its resource-intensive nature. Innovative approaches, such as Mixture of Prompts (MoPs) to manage heterogeneous tasks and data distributions, and frameworks like Prompt Middleware for streamlined prompt generation, should be explored. By tackling these issues, we can significantly improve output quality and user satisfaction, leading to more efficient and adaptable LLM applications [50, 72, 17]. Developing automated and scalable approaches to prompt design will enable the realization of LLMs' full potential across diverse domains.

6.4 Bias and Ethical Concerns

Bias and ethical concerns are critical considerations in deploying LLMs across various domains. A significant issue is the potential for LLMs to produce erroneous results or 'hallucinations', undermining data retrieval reliability [2]. Privacy concerns arise when LLMs are prompted with examples containing sensitive information, necessitating fairness to prevent bias perpetuation in AI outputs [3]. The lack of transparency in LLM prompt interpretation exacerbates these issues, leading to biased or erroneous code generation and complicating ethical AI practices [23].

In educational contexts, AI-generated content quality raises concerns about fostering academic dishonesty, undermining educational integrity [5]. Additionally, the potential for LLMs to generate harmful content necessitates careful examination of ethical deployment implications, as current studies often fall short in addressing these critical aspects [6].

Limitations in handling non-visual states and the potential for generating ungrounded descriptions can introduce biases and distractions in AI outputs, particularly when integrating visual and textual data [21]. These challenges underscore the importance of developing comprehensive governance frameworks to ensure equitable and responsible LLM use, addressing biases and ethical concerns to safeguard the integrity and trustworthiness of AI technologies.

6.5 Scalability and Generalization

Scalability and generalization present significant challenges in deploying LLMs and prompt programming, particularly as these technologies are applied across diverse domains and tasks. The complexity of sequential decision-making in dynamic environments can impede scalability, as highlighted by difficulties in planning tasks [98]. This complexity is exacerbated by heterogeneous data distributions, which can affect LLM effectiveness in varied scenarios. The generalizability of LLM applications is often constrained by the context-specific nature of many techniques, which may not perform consistently across all tasks or models [4]. This limitation is evident in studies focusing on specific languages or datasets, posing challenges for generalizing findings to broader applications [37].

Integrating LLMs into diverse application domains raises concerns about scalability and generalization. Current methods validated primarily in simulations may not translate effectively to physical hardware, underscoring the need for comprehensive evaluation frameworks that address these gaps. Additionally, the ethical implications of LLM-based recommendations necessitate systematic studies to ensure responsible deployment [99]. Benchmarks designed to evaluate LLM performance often face scalability limitations due to resource constraints, preventing coverage of all conversational scenarios or larger model experiments [100]. Furthermore, the assumption of prior knowledge of harmful outputs in some benchmarks limits their effectiveness, as this is often not feasible in real-world applications.

To address these challenges, future research should focus on adapting detection methods to evolving adversarial strategies and improving the handling of false positives and negatives, thereby enhancing LLM robustness and generalizability. By creating more scalable and generalizable frameworks, researchers can unlock the full potential of LLMs, facilitating effective deployment in complex real-world applications, from scientific communication and programming tasks to improving safety and alignment in AI systems. This approach will not only address current limitations such as biases and inaccuracies but also ensure reliable LLM integration into diverse fields, including healthcare and software engineering, maximizing their utility and impact [49, 75, 101, 102, 6].

7 Conclusion

7.1 Future Research Directions

The future of prompt programming and large language models (LLMs) lies in addressing key areas that promise to enhance their capabilities and broaden their application across diverse fields. A significant focus should be on refining the stability and accuracy of Inductive Bias Learning (IBL), which not only offers a potential alternative to conventional machine learning algorithms but also provides deeper insights into the logic underpinning code generation. Enhancing models like MANIPLE by incorporating a variety of fact types can further improve their adaptability and performance in different programming scenarios, thereby expanding the utility of LLMs in computational tasks.

In the educational sector, research should prioritize the formulation of ethical guidelines for LLM usage, alongside the development of novel assessment methodologies to evaluate the impact of LLM integration on student engagement and learning outcomes over time. Such efforts are crucial to ensure that LLMs are used responsibly and effectively, fostering beneficial educational experiences.

Improving the efficiency of LLMs and exploring innovative training paradigms are critical areas for research, particularly in addressing ethical issues associated with their deployment to ensure alignment with societal norms and expectations. Moreover, enhancing the grounding of LLM-generated descriptions in visual data and improving their handling of non-visual concepts could significantly boost their performance in complex, multi-modal tasks.

Further research into optimizing the prompt learning process and exploring various coupling strategies within frameworks like MaPLe could elevate LLM performance in vision-language tasks. Extending the application of MaPLe to other vision-language models or tasks presents a promising avenue for developing more robust and versatile AI systems.

By pursuing these research directions, the field of prompt programming and LLMs can continue to evolve, driving innovation and expanding the scope of AI technology across a multitude of complex domains.

References

[1] Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. Openprompt: An open-source framework for prompt-learning. *arXiv preprint arXiv:2111.01998*, 2021.

[2] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. Querying large language models with sql, 2023.

[3] Cheng Qian, Chi Han, Yi R. Fung, Yujia Qin, Zhiyuan Liu, and Heng Ji. Creator: Tool creation for disentangling abstract and concrete reasoning of large language models, 2024.

[4] Orion Weller, Marc Marone, Nathaniel Weir, Dawn Lawrie, Daniel Khashabi, and Benjamin Van Durme. "according to ...": Prompting language models improves quoting from pre-training data, 2024.

[5] James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Peterson, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. The robots are here: Navigating the generative ai revolution in computing education, 2023.

[6] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*, 2023.

[7] Rajdeep Mondal, Alan Tang, Ryan Beckett, Todd Millstein, and George Varghese. What do llms need to synthesize correct router configurations?, 2023.

[8] Simran Arora, Avanika Narayan, Mayee F. Chen, Laurel Orr, Neel Guha, Kush Bhatia, Ines Chami, Frederic Sala, and Christopher Ré. Ask me anything: A simple strategy for prompting language models, 2022.

[9] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm, 2021.

[10] Grant Oosterwyk, Pitso Tsibolane, Popyeni Kautondokwa, and Ammar Canani. Beyond the hype: A cautionary tale of chatgpt in the programming classroom, 2024.

[11] Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: A case study. In *International Conference on Machine Learning*, pages 41092–41110. PMLR, 2023.

[12] Zhuofeng Wu, He Bai, Aonan Zhang, Jiatao Gu, VG Vinod Vydiswaran, Navdeep Jaitly, and Yizhe Zhang. Divide-or-conquer? which part should you distill your llm?, 2024.

[13] Zangir Iklassov, Yali Du, Farkhad Akimov, and Martin Takac. Self-guiding exploration for combinatorial problems, 2024.

[14] Eason Chen. Generate labeled training data using prompt programming and gpt-3. an example of big five personality classification, 2023.

[15] Jenny T. Liang, Melissa Lin, Nikitha Rao, and Brad A. Myers. Prompts are programs too! understanding how developers build software containing prompts, 2024.

[16] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[17] Stephen MacNeil, Andrew Tran, Joanne Kim, Ziheng Huang, Seth Bernstein, and Dan Mogil. Prompt middleware: Mapping prompts for large language models to ui affordances, 2023.

[18] Li Zhang, Liam Dugan, Hainiu Xu, and Chris Callison-Burch. Exploring the curious case of code prompts, 2023.

[19] James Prather, Paul Denny, Juho Leinonen, David H. Smith IV au2, Brent N. Reeves, Stephen MacNeil, Brett A. Becker, Andrew Luxton-Reilly, Thezyrie Amarouche, and Bailey Kimmel. Interactions with prompt problems: A new way to teach programming with large language models, 2024.

[20] Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36:51008–51025, 2023.

[21] Wentao Bao, Lichang Chen, Heng Huang, and Yu Kong. Prompting language-informed distribution for compositional zero-shot learning, 2024.

[22] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122, 2023.

[23] Zhenlan Ji, Pingchuan Ma, Zongjie Li, and Shuai Wang. Benchmarking and explaining large language model-based code generation: A causality-centric approach, 2023.

[24] Pengshuo Qiu, Frank Rudzicz, and Zining Zhu. Scenarios and approaches for situated natural language explanations, 2024.

[25] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, 36:21558–21572, 2023.

[26] Nikhil Parasaram, Huijie Yan, Boyu Yang, Zineb Flahy, Abriele Qudsi, Damian Ziaber, Earl Barr, and Sergey Mechtaev. The fact selection problem in llm-based program repair, 2024.

[27] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[28] Xinbei Ma, Tianjie Ju, Jiyang Qiu, Zhuosheng Zhang, Hai Zhao, Lifeng Liu, and Yulong Wang. On the robustness of editing large language models, 2024.

[29] Liam Hazan, Gili Focht, Naama Gavrielov, Roi Reichart, Talar Hagopian, Mary-Louise C. Greer, Ruth Cytter Kuint, Dan Turner, and Moti Freiman. Leveraging prompt-learning for structured information extraction from crohn's disease radiology reports in a low-resource language, 2024.

[30] Aleksandra Edwards and Jose Camacho-Collados. Language models for text classification: Is in-context learning enough?, 2024.

[31] Priti Oli, Rabin Banjade, Jeevan Chapagain, and Vasile Rus. The behavior of large language models when prompted to generate code explanations, 2023.

[32] Florian Tambon, Amin Nikanjam, Foutse Khomh, and Giuliano Antoniol. Assessing programming task difficulty for efficient evaluation of large language models, 2024.

[33] Xu Guo and Yiqiang Chen. Generative ai for synthetic data generation: Methods, challenges and the future, 2024.

[34] Jinxin Liu, Shulin Cao, Jiaxin Shi, Tingjian Zhang, Lunyiu Nie, Linmei Hu, Lei Hou, and Juanzi Li. How proficient are large language models in formal languages? an in-depth insight for knowledge base question answering, 2024.

[35] John X. Morris, Chandan Singh, Alexander M. Rush, Jianfeng Gao, and Yuntian Deng. Tree prompting: Efficient task adaptation without fine-tuning, 2023.

[36] David Wingate, Mohammad Shoeybi, and Taylor Sorensen. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*, 2022.

[37] Jane Pan, Tianyu Gao, Howard Chen, and Danqi Chen. What in-context learning "learns" in-context: Disentangling task recognition and task learning, 2023.

[38] Hideki Deguchi, Kazuki Shibata, and Shun Taguchi. Language to map: Topological map generation from natural language path instructions, 2024.

[39] Milena Chadimová, Eduard Jurášek, and Tomáš Kliegr. Meaningless is better: hashing bias-inducing words in llm prompts improves performance in logical reasoning and statistical learning, 2024.

[40] Philip Feldman, James R. Foulds, and Shimei Pan. Trapping llm hallucinations using tagged context prompts, 2023.

[41] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models. In *International Conference on Machine Learning*, pages 10764–10799. PMLR, 2023.

[42] Hongwei Jin, George Papadimitriou, Krishnan Raghavan, Pawel Zuk, Prasanna Balaprakash, Cong Wang, Anirban Mandal, and Ewa Deelman. Large language models for anomaly detection in computational workflows: from supervised fine-tuning to in-context learning, 2024.

[43] Yuan Tian, Weiwei Cui, Dazhen Deng, Xinjing Yi, Yurun Yang, Haidong Zhang, and Yingcai Wu. Chartgpt: Leveraging llms to generate charts from abstract natural language, 2023.

[44] Sangwon Yu, Changmin Lee, Hojin Lee, and Sungroh Yoon. Controlled text generation for black-box language models via score-based progressive editor, 2024.

[45] Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. Autoflow: Automated workflow generation for large language model agents, 2024.

[46] Yankai Zeng, Abhiramon Rajashekharan, Kinjal Basu, Huaduo Wang, Joaquín Arias, and Gopal Gupta. A reliable common-sense reasoning socialbot built using llms and goal-directed asp, 2024.

[47] Toma Tanaka, Naofumi Emoto, and Tsukasa Yumibayashi. Inductive-bias learning: Generating code models with large language model, 2023.

[48] Lincoln Murr, Morgan Grainger, and David Gao. Testing llms on code generation with varying levels of prompt specificity, 2023.

[49] Yi Luo, Zhenghao Lin, Yuhao Zhang, Jiashuo Sun, Chen Lin, Chengjin Xu, Xiangdong Su, Yelong Shen, Jian Guo, and Yeyun Gong. Ensuring safe and high-quality outputs: A guideline library approach for language models, 2024.

[50] Benfeng Xu, An Yang, Junyang Lin, Quan Wang, Chang Zhou, Yongdong Zhang, and Zhendong Mao. Expertprompting: Instructing large language models to be distinguished experts. *arXiv preprint arXiv:2305.14688*, 2023.

[51] Yingzhe Peng, Xiaoting Qin, Zhiyang Zhang, Jue Zhang, Qingwei Lin, Xu Yang, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. Navigating the unknown: A chat-based collaborative interface for personalized exploratory tasks, 2024.

[52] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[53] Zhenyu Wu, Meng Jiang, and Chao Shen. Get an a in math: Progressive rectification prompting, 2023.

[54] Andrew Bell and Joao Fonseca. Output scouting: Auditing large language models for catastrophic responses, 2024.

[55] Roma Shusterman, Allison C. Waters, Shannon O'Neill, Phan Luu, and Don M. Tucker. An active inference strategy for prompting reliable responses from large language models in medical practice, 2024.

[56] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T Joshi, Hanna Moazam, et al. Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*, 2023.

[57] Mozes van de Kar, Mengzhou Xia, Danqi Chen, and Mikel Artetxe. Don't prompt, search! mining-based zero-shot learning with language models, 2022.

[58] C. Nicolò De Sabbata, Theodore R. Sumers, and Thomas L. Griffiths. Rational metareasoning for large language models, 2024.

[59] Daniel Khashabi, Shane Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, and Yejin Choi. Prompt waywardness: The curious case of discretized interpretation of continuous prompts, 2022.

[60] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*, 2023.

[61] Gaurav Sahu, Olga Vechtomova, Dzmitry Bahdanau, and Issam H. Laradji. Promptmix: A class boundary augmentation method for large language model distillation, 2023.

[62] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*, 2024.

[63] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, 2022.

[64] Haoyu Gao, Ting-En Lin, Hangyu Li, Min Yang, Yuchuan Wu, Wentao Ma, and Yongbin Li. Self-explanation prompting improves dialogue understanding in large language models, 2023.

[65] Ziyang Chen and Stylios Moscholios. Using prompts to guide large language models in imitating a real person's language style, 2024.

[66] Yanxin Chen and Ling He. Research on the application of large language models in automatic question generation: A case study of chatglm in the context of high school information technology curriculum, 2024.

[67] Sakshi Mahendru and Tejul Pandit. Venn diagram prompting : Accelerating comprehension with scaffolding effect, 2024.

[68] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*, 2022.

[69] Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. The impact of reasoning step length on large language models, 2024.

[70] Yajing Wang and Zongwei Luo. Enhance multi-domain sentiment analysis of review texts through prompting strategies, 2024.

[71] Mandana Vaziri, Louis Mandel, Claudio Spiess, and Martin Hirzel. Pdl: A declarative prompt programming language, 2024.

[72] Chen Dun, Mirian Hipolito Garcia, Guoqing Zheng, Ahmed Hassan Awadallah, Anastasios Kyrillidis, and Robert Sim. Sweeping heterogeneity with smart mops: Mixture of prompts for llm task adaptation, 2023.

[73] Imen Azaiz, Natalie Kiesler, and Sven Strickroth. Feedback-generation for programming exercises with gpt-4, 2024.

[74] Mengdi Xu, Peide Huang, Wenhao Yu, Shiqi Liu, Xilun Zhang, Yaru Niu, Tingnan Zhang, Fei Xia, Jie Tan, and Ding Zhao. Creative robot tool use with large language models, 2023.

[75] Jan Clusmann, Fiona R Kolbinger, Hannah Sophie Muti, Zunamys I Carrero, Jan-Niklas Eckardt, Narmin Ghaffari Laleh, Chiara Maria Lavinia Löffler, Sophie-Caroline Schwarzkopf, Michaela Unger, Gregory P Veldhuizen, et al. The future landscape of large language models in medicine. *Communications medicine*, 3(1):141, 2023.

[76] Wenxiao Zhang, Xiangrui Kong, Conan Dewitt, Thomas Braunl, and Jin B. Hong. A study on prompt injection attack against llm-integrated mobile robotic systems, 2024.

[77] Anqi Wang, Zhizhuo Yin, Yulu Hu, Yuanyuan Mao, and Pan Hui. Exploring the potential of large language models in artistic creation: Collaboration and reflection on creative programming, 2024.

[78] Pei-Fu Guo, Ying-Hsuan Chen, Yun-Da Tsai, and Shou-De Lin. Towards optimizing with large language models, 2024.

[79] Zhiqiang Wang, Yiran Pang, and Yanbin Lin. Large language models are zero-shot text classifiers, 2023.

[80] Ekaterina Svikhnushina and Pearl Pu. Approximating online human evaluation of social chatbots with prompting, 2023.

[81] Lars Krupp, Jonas Bley, Isacco Gobbi, Alexander Geng, Sabine Müller, Sungho Suh, Ali Moghiseh, Arcesio Castaneda Medina, Valeria Bartsch, Artur Widera, Herwig Ott, Paul Lukowicz, Jakob Karolus, and Maximilian Kiefer-Emmanouilidis. Llm-generated tips rival expert-created tips in helping students answer quantum-computing questions, 2024.

[82] Tanushree Banerjee, Richard Zhu, Runzhe Yang, and Karthik Narasimhan. Llms are superior feedback providers: Bootstrapping reasoning for lie detection with self-generated feedback, 2024.

[83] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[84] Yue Zhang, Hehe Fan, and Yi Yang. Prompt-aware adapter: Towards learning adaptive visual tokens for multimodal large language models, 2024.

[85] Zifeng Wang, Cao Xiao, and Jimeng Sun. Autotrial: Prompting language models for clinical trial design, 2023.

[86] Daniil Filienko, Yinzhou Wang, Caroline El Jazmi, Serena Xie, Trevor Cohen, Martine De Cock, and Weichao Yuwen. Toward large language models as a therapeutic tool: Comparing prompting techniques to improve gpt-delivered problem-solving therapy, 2024.

[87] An Quang Tang, Xiuzhen Zhang, and Minh Ngoc Dinh. Ignitioninnovators at "discharge me!": Chain-of-thought instruction finetuning large language models for discharge summaries, 2024.

[88] Daoguang Zan, Ailun Yu, Wei Liu, Dong Chen, Bo Shen, Wei Li, Yafen Yao, Yongshun Gong, Xiaolin Chen, Bei Guan, Zhiguang Yang, Yongji Wang, Qianxiang Wang, and Lizhen Cui. Codes: Natural language to code repository via multi-layer sketch, 2024.

[89] Sidong Feng and Chunyang Chen. Prompting is all you need: Automated android bug replay with large language models, 2024.

[90] Lukas Netz, Jan Reimer, and Bernhard Rumpe. Using grammar masking to ensure syntactic validity in llm-based modeling tasks, 2024.

[91] Gustavo Pinto, Cleidson de Souza, João Batista Neto, Alberto de Souza, Tarcísio Gotto, and Edward Monteiro. Lessons from building stackspot ai: A contextualized ai coding assistant, 2024.

[92] Jules White, Sam Hays, Quchen Fu, Jesse Spencer-Smith, and Douglas C Schmidt. Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative AI for Effective Software Development*, pages 71–108. Springer, 2024.

[93] Dong Shu and Mengnan Du. Comparative analysis of demonstration selection algorithms for llm in-context learning, 2024.

[94] Ken Gu, Ruoxi Shang, Tim Althoff, Chenglong Wang, and Steven M. Drucker. How do analysts understand and verify ai-assisted data analyses?, 2024.

[95] Chengcheng Han, Xiaowei Du, Che Zhang, Yixin Lian, Xiang Li, Ming Gao, and Baoyuan Wang. Dialcot meets ppo: Decomposing and exploring reasoning paths in smaller language models, 2023.

[96] Harshit Joshi, José Cambronero Sanchez, Sumit Gulwani, Vu Le, Gust Verbruggen, and Ivan Radiček. Repair is nearly generation: Multilingual program repair with llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5131–5140, 2023.

[97] Zhanming Jie and Wei Lu. Leveraging training data in few-shot prompting for numerical reasoning, 2023.

[98] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pages 1–7, 2021.

[99] Bocheng Chen, Hanqing Guo, Guangjing Wang, Yuanda Wang, and Qiben Yan. The dark side of human feedback: Poisoning large language models via user inputs, 2024.

[100] Sander Land and Max Bartolo. Fishing for magikarp: Automatically detecting under-trained tokens in large language models, 2024.

[101] Javier González and Aditya V. Nori. Beyond words: A mathematical framework for interpreting large language models, 2023.

[102] Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey, 2024.

Disclaimer:

SurveyX is an AI-powered system designed to automate the generation of surveys. While it aims to produce high-quality, coherent, and comprehensive surveys with accurate citations, the final output is derived from the AI's synthesis of pre-processed materials, which may contain limitations or inaccuracies. As such, the generated content should not be used for academic publication or formal submissions and must be independently reviewed and verified. The developers of SurveyX do not assume responsibility for any errors or consequences arising from the use of the generated surveys.