A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications

Pranab Sahoo[1] , **Ayush Kumar Singh**[1] , **Sriparna Saha**[1] , **Vinija Jain**[2,3*] , **Samrat Mondal**[1] and **Aman Chadha**[2,3*]
[1]Department of Computer Science And Engineering, Indian Institute of Technology Patna
[2]Stanford University, [3]Amazon AI
{pranab_2021cs25, ayush_2211ai27, sriparna, samrat}@iitp.ac.in, hi@vinija.ai, hi@aman.ai

Abstract

Prompt engineering has emerged as an indispensable technique for extending the capabilities of large language models (LLMs) and vision-language models (VLMs). This approach leverages task-specific instructions, known as prompts, to enhance model efficacy without modifying the core model parameters. Rather than updating the model parameters, prompts allow seamless integration of pre-trained models into downstream tasks by eliciting desired model behaviors solely based on the given prompt. Prompts can be natural language instructions that provide context to guide the model or learned vector representations that activate relevant knowledge. This burgeoning field has enabled success across various applications, from question-answering to commonsense reasoning. However, there remains a lack of systematic organization and understanding of the diverse prompt engineering methods and techniques. This survey paper addresses the gap by providing a structured overview of recent advancements in prompt engineering, categorized by application area. For each prompting approach, we provide a summary detailing the prompting methodology, its applications, the models involved, and the datasets utilized. We also delve into the strengths and limitations of each approach and include a taxonomy diagram and table summarizing datasets, models, and critical points of each prompting technique. This systematic analysis enables a better understanding of this rapidly developing field and facilitates future research by illuminating open challenges and opportunities for prompt engineering.

1 Introduction

LLM과 VLM의 Prompting을 정리

Prompt engineering has emerged as a crucial technique for enhancing the capabilities of pre-trained large language models (LLMs) and vision-language models (VLMs). It involves strategically designing task-specific instructions, referred to as prompts, to guide model output without altering parameters. The significance of prompt engineering is especially evident in its transformative impact on the adaptability of LLMs and VLMs. By offering a mechanism to fine-tune model outputs through carefully crafted instructions, prompt engineering enables these models to excel across diverse tasks and domains. This adaptability is different from traditional paradigms, where model retraining or extensive fine-tuning is often required for task-specific performance. This is the transformative promise of prompt engineering, pushing the boundaries of AI and opening doors to a future brimming with possibilities. In an ever-evolving landscape, ongoing research consistently reveals innovative approaches and applications within prompt engineering. The significance of prompt engineering is underscored by its capacity to steer model responses, enhancing the adaptability and applicability of LLMs across diverse sectors. The landscape of contemporary prompt engineering spans a spectrum of techniques, encompassing foundational methods like zero-shot and few-shot prompting to more intricate approaches such as "chain of code" prompting. The notion of prompt engineering was initially investigated and popularized in the LLMs [Liu *et al.*, 2023], [Tonmoy *et al.*, 2024], [Chen *et al.*, 2023] later extended to VLMs [Wu *et al.*, 2023], [Bahng *et al.*, 2022]. Despite the extensive literature on prompt engineering within both LLMs and VLMs, a notable gap remains, particularly concerning a systematic overview of application-centric prompt engineering techniques. With recent strides in prompt engineering, there is a pressing need for a comprehensive survey that offers a nuanced understanding of applications and advancements in contemporary research. This survey dives deep into the ever-evolving landscape of prompt engineering, analyzing over 41 distinct techniques categorized by their diverse applications. Employing a systematic

~로 가득차다

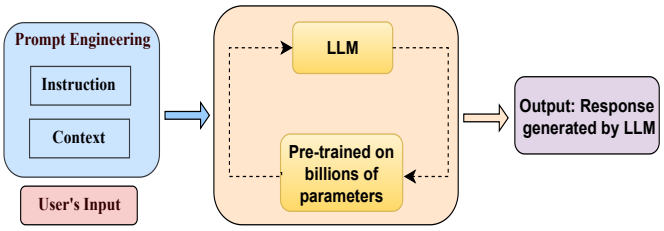

Figure 1: Visual breakdown of prompt engineering components: LLMs trained on extensive data, instruction and context as pivotal elements shaping the prompt, and a user input interface.

*Work does not relate to position at Amazon.

review approach, we meticulously delve into the intricacies of diverse cutting-edge prompting methods. Our examination encompasses their applications, the language models utilized, and the datasets subjected to experimentation, providing a detailed and nuanced analysis of the evolving landscape of prompt engineering. Additionally, we discuss the pros and cons of these techniques, offering insights into their comparative efficacy. We present a comprehensive taxonomy diagram that illustrates how these techniques navigate the vast landscape of LLM capabilities (see Fig.2) and provide a table summarizing the datasets, employed models, and evaluation metrics (see Table1). From language generation and question answering to code creation and reasoning tasks, prompt engineering empowers the LLMs into performing feats we never thought possible. By bridging the existing gap in the literature, this survey aims to serve as a valuable resource for researchers and practitioners, offering insights into the latest developments and facilitating a deeper understanding of the evolving landscape of prompt engineering. The structure of the paper is organized as follows: Section 2 presents the prompt engineering techniques from both basic to advanced by categorizing application-area and Section 3 provides a conclusion along with considerations for future research endeavors.

2 Prompt Engineering

In this section, we have organized prompt engineering techniques according to their application areas and provided a concise overview of the evolution of prompting techniques, spanning from zero-shot prompting to the latest advancements.

2.1 New Tasks Without Extensive Training

Zero-Shot Prompting

Zero-shot prompting offers a paradigm shift in leveraging large LLMs. This technique removes the need for extensive training data, instead relying on carefully crafted prompts that guide the model toward novel tasks [Radford *et al.*, 2019]. Specifically, the model receives a task description in the prompt but lacks labeled data for training on specific input-output mappings. The model then leverages its pre-existing knowledge to generate predictions based on the given prompt for the new task.

학습 데이터 없이 프롬프트 만으로 새로운 Task를 수행하도록 유도함

Few-Shot Prompting

Few-shot prompting provides models with a few input-output examples to induce an understanding of a given task, unlike zero-shot prompting, where no examples are supplied [Brown *et al.*, 2020]. Providing even a few high-quality examples has improved model performance on complex tasks compared to no demonstration. However, few-shot prompting requires additional tokens to include the examples, which may become prohibitive for longer text inputs. Moreover, the selection and composition of prompt examples can significantly influence model behavior, and biases like favoring frequent words may still affect few-shot results. While few-shot prompting enhances capabilities for complex tasks, especially among large pre-trained models like GPT-3, careful prompt engineering is critical to achieve optimal performance and mitigate unintended model biases.

금지하는

Fewshot은 Input - Output Example을 제공하는 방법
- 추가 토큰이 요구되어 긴 Input Context에서 성능이 저하되기도 함
- Example의 선택과 구성은 모델에 큰 영향을 끼침
- 다빈도의 토큰이 있으면 Bias가 생길 수 있음

CoT는 일관된, 단계별 추론 과정을 제공하는 방법
논리적 추론 과정을 전달함으로써,
주어진 프롬프트를 더 깊이있게 이해시킴

2.2 Reasoning and Logic

Chain-of-Thought (CoT) Prompting

LLMs often stumble in the face of complex reasoning, limiting their potential. Aiming to bridge this gap, Wei *et al.* [2022] introduced Chain-of-Thought (CoT) prompting as a technique to prompt LLMs in a way that facilitates coherent and step-by-step reasoning processes. The primary contribution lies in the proposal and exploration of CoT prompting, demonstrating its effectiveness in eliciting more structured and thoughtful responses from LLMs compared to traditional prompts. Through a series of experiments, the authors showcase the distinctive qualities of CoT prompting, emphasizing its ability to guide LLMs through a logical reasoning chain. This results in responses that reflect a deeper understanding of the given prompts. For example, the prompt would show the reasoning process and final answer for a multi-step math word problem and mimic how humans break down problems into logical intermediate steps. The authors achieved state-of-the-art performance in math and commonsense reasoning benchmarks by utilizing CoT prompts for PaLM 540B, achieving an accuracy of 90.2%.

Automatic Chain-of-Thought (Auto-CoT) Prompting

Manual creation of high-quality CoT examples is both time-consuming and suboptimal. Zhang *et al.* [2022] introduced Auto-CoT to automatically instruct LLMs with a "Let's think step-by-step" prompt to generate reasoning chains. Recognizing the possibility of errors in individually generated chains, Auto-CoT enhances robustness through diverse sampling. It samples various questions and generates multiple distinct reasoning chains for each, forming a final set of demonstrations. This automated diverse sampling minimizes errors and enhances few-shot learning, eliminating the need for labor-intensive manual creation of reasoning chains. Auto-CoT demonstrated enhanced performance, surpassing the CoT paradigm with average accuracy improvements of 1.33% and 1.5% on arithmetic and symbolic reasoning tasks, respectively, employing GPT-3.

CoT는 실수도 많고 시간도 많이 걸림
"Let's thkin step-by-step" 등을 프롬프트에 추가하여
자율적으로 Reasoning Chain을 수행하게 함

Self-Consistency

Wang *et al.* [2022] introduced self-consistency, a decoding strategy enhancing reasoning performance compared to greedy decoding in CoT prompting. For complex reasoning tasks with multiple valid paths, self-consistency generates diverse reasoning chains by sampling from the language model's decoder. It then identifies the most consistent final answer by marginalizing these sampled chains. This approach capitalizes on the observation that problems requiring thoughtful analysis often entail greater reasoning diversity, leading to a solution. The combination of self-consistency and chain-of-thought prompting results in significant accuracy improvements across various benchmarks, such as 17.9% on GSM8K, 11.0% on SVAMP, 12.2% on AQuA, 6.4% on StrategyQA, and 3.9% on ARC-challenge compared to the baseline chain-of-thought prompting.

다양한 Reasoning Chain을 샘플링한 후,
가장 일관된 최종 답안을 확률적으로 도출해내는 방법

Logical Chain-of-Thought (LogiCoT) Prompting

The ability to perform logical reasoning is critical for LLMs to solve complex, multi-step problems across diverse domains. Existing methods, like CoT prompting, encourage step-by-step

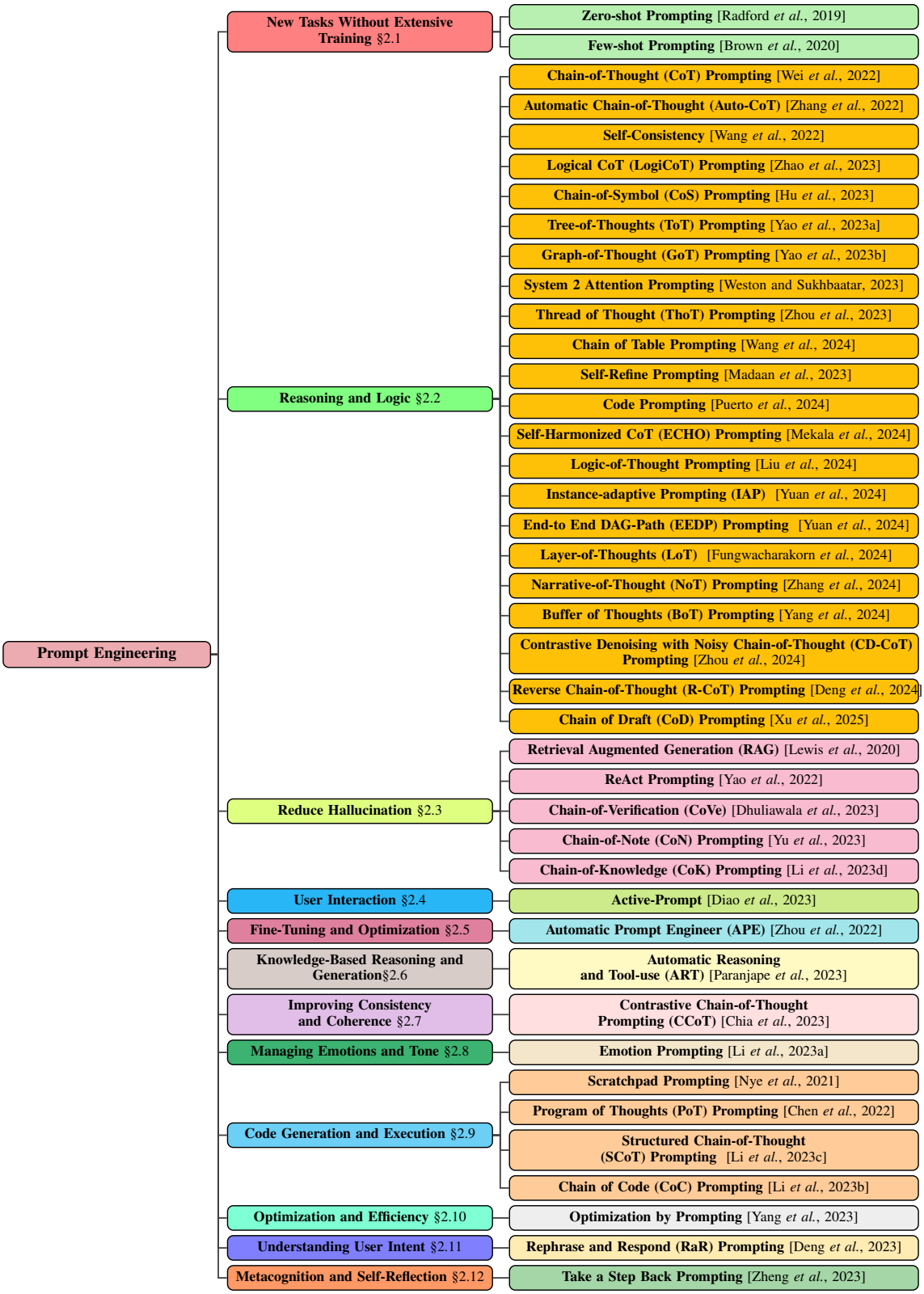

Figure 2: Taxonomy of prompt engineering techniques in LLMs, organized around application domains, providing a nuanced framework for customizing prompts across diverse contexts.

LogiCoT는 Step-by-step Reasoning Chain에 verification을 추가한 것
모델이 생성한 각 추론 단계를 검증하고 잘못된 Step에 대한 피드백을 제공하여 정정하게 함
생각-검증-수정의 순환과정을 구현하여 논리적오류와 할루시네이션을 감소시킴

reasoning but lack effective verification mechanisms. Zhao *et al.* [2023] proposes a Logical Chain-of-Thought (LogiCoT) prompting, a neurosymbolic framework that leverages principles from symbolic logic to enhance reasoning in a coherent and structured manner. Specifically, LogiCoT applies the concept of reductio ad absurdum to verify each step of reasoning generated by the model and provide targeted feedback to revise incorrect steps. LogiCoT can reduce logical errors and hallucinations through a think-verify-revise loop. Experimenting with Vicuna-33b and GPT-4, the findings underscore LogiCoT's notable enhancement of reasoning abilities, exhibiting improvements of 0.16% and 1.42% on the GSM8K dataset and 3.15% and 2.75% on the AQuA dataset compared to CoT, respectively.

Chain-of-Symbol (CoS) Prompting

LLMs often struggle with tasks involving complex spatial relationships due to their reliance on natural language, which is susceptible to ambiguity and biases. To overcome this limitation, Hu *et al.* [2023] introduced CoS, employing condensed symbols instead of natural language. CoS provides distinct advantages: clear and concise prompts, heightened spatial reasoning for LLMs, and improved human interpretability. CoS suffers from challenges such as scalability, generalizability, integration with other techniques, and interpretability of LLM reasoning based on symbols. Notably, the implementation of CoS significantly elevates ChatGPT's performance, boosting accuracy from 31.8% to an impressive 92.6% on Brick World tasks. Moreover, CoS achieves up to a 65.8% reduction in prompt tokens, streamlining the process while maintaining high accuracy.

LLM의 공간적 관계 요구 작업을 개선을 위해 제안
자연어 대신 압축된 기호를 사용함

Tree-of-Thoughts (ToT) Prompting

CoT의 중간 Step의 사고(thought)들을 Tree 형태로 관리
thought는 최종 해답을 향해 나아가는 일관된 시퀀스를 의미.
모델은 이를 기반으로 현재까지의 추론 진행상황을 평가함

BFS, DFS와 결합되어 다양한 추론 경로를 체계적으로 탐색하고 역추적할 수 있도록 함

Yao *et al.* [2023a] and Long [2023] proposed the Tree-of-Thoughts (ToT) framework to enhance prompting capabilities for complex tasks requiring exploration and look-ahead reasoning. ToT extends CoT prompting by managing a tree structure of intermediate reasoning steps, known as "thoughts". Each thought represents a coherent language sequence moving toward the final solution. This structure allows language models to deliberately reason by assessing the progress generated by thoughts in solving the problem. ToT integrates the model's abilities to produce and evaluate thoughts with search algorithms like breadth-first or depth-first search. This enables systematic exploration among reasoning chains, with a look-ahead to expand promising directions and to backtrack when solutions are incorrect. ToT excelled in the Game of 24 tasks, achieving a 74% success rate compared to CoT's 4%. Additionally, in word-level tasks, ToT outperformed CoT with a 60% success rate versus 16%.

Graph-of-Thoughts (GoT) Prompting

The inherent non-linear nature of human thought processes challenges the conventional sequential approach of CoT prompting. Yao *et al.* [2023b] introduced the "Graph of Thoughts" prompting, a graph-based framework advancing traditional sequential methods to better align with the non-linear characteristics of human thinking. This framework permits dynamic interplay, backtracking, and evaluation of ideas, allowing the aggregation and combination of thoughts from various branches, departing from the linear structure of the tree of thoughts. The key contributions encompass modeling the reasoning process as a directed graph, offering a modular architecture with diverse transformation operations. The framework is presented as a versatile and dynamic approach to language model prompting, capturing the intricacies of human thought processes and enhancing model capabilities. The GoT reasoning model demonstrates substantial gains over the CoT baseline, improving accuracy by 3.41% with T5-base and 5.08% with T5-large on GSM8K. It also boosts accuracy over the state-of-the-art Multimodal-CoT by 6.63% using T5-base and 1.09% with T5-large on ScienceQA.

상호작용

방향성 Graph로 사고들을 묶는 방법
다양한 Branch로 부터 사고들을 합치고,
생각들을 동적 상호작용, 역추적, 평가할 수 있게 함

2 단계로 생성
1) Input Context를 재생성하여 필요한 정보만 정제
2) 응답 생성

System 2 Attention (S2A) Prompting

The soft attention mechanism in Transformer-based LLMs is prone to incorporating irrelevant context information, impacting token generation adversely. To address this, Weston and Sukhbaatar [2023] proposed System 2 Attention (S2A), utilizing the reasoning abilities of LLMs to selectively attend to relevant portions by regenerating the input context. S2A employs a two-step process to enhance attention and response quality by employing context regeneration and response generation with refined context. The effectiveness of S2A is evaluated across various tasks, including factual QA, long-form generation, and math word problems. In factual QA, S2A attains an accuracy of 80.3%, demonstrating a substantial enhancement in factuality. In long-form generation, it improves objectivity and receives a score of 3.82 out of 5.

1) 전체 맥락을 Segment 단위로 요약하고 각 부분을 정밀하게 검토함
2) 분석된 정보를 통합/정제하여 최종 응답 생성

Thread of Thought (ThoT) Prompting

Zhou *et al.* [2023] presented Thread of Thought (ThoT), a prompting technique designed to enhance the reasoning abilities of LLMs within chaotic contexts. ThoT, inspired by human cognition, systematically examines extensive contexts into manageable segments for incremental analysis, employing a two-phase approach where the LLM first summarizes and examines each segment before refining the information for a final response. ThoT's flexibility shines as a versatile "plug-and-play" module, enhancing reasoning across different models and prompting methods. Evaluations on question answering and conversation datasets reveal substantial performance improvements of 47.20% and 17.8%, respectively, especially in chaotic contexts.

단계별 Table 추론을 중심으로 SQL / DataFrame 연산을 동적으로 생성/실행
반복적인 수행 과정을 통해 중간 결과를 계속 정제함

Chain-of-Table Prompting

Approaches like CoT, PoT, and ToT represent reasoning steps through free-form text or code, which face challenges when dealing with intricate table scenarios. The study by Wang *et al.* [2024] introduced a pioneering prompting technique named Chain-of-Table. This method uses step-by-step tabular reasoning by dynamically generating and executing common SQL/-DataFrame operations on tables. The iterative nature of this process enhances intermediate results, empowering LLMs to make predictions through logically visualized reasoning chains. Significantly, Chain-of-Table consistently improves the performance of two benchmark tabular datasets by 8.69% on TabFact and 6.72% on WikiTQ, respectively.

Self-Refine Prompting

Self-Refine prompting, proposed by Madaan *et al.* [2023], enhances LLM performance by iteratively refining outputs

LLM의 초기 답변은 복잡하거나 모호한 목표, 다단계 Reasoning을 잘 수행 못함
1) 초기 응답을 생성
2) 스스로 비판적 분석을 수행
3) 피드백을 바탕으로 응답을 정제함

through self-generated feedback, mimicking human revision. While LLMs can handle a wide range of tasks, they often struggle with complex objectives, ambiguous goals, or multi-step reasoning, leading to initial responses with inaccuracies or flawed logic. Inspired by human iterative refinement, Self-Refine enables LLMs to improve their outputs through a structured three-step process: generating an initial response, prompting the model to critique its own output, and refining the response based on this feedback. This cycle continues until predefined stopping criteria are met, allowing the model to produce more accurate and contextually relevant results. Unlike traditional prompting methods, which rely solely on a single-step response, Self-Refine fosters incremental improvement, making it particularly effective for tasks requiring nuanced reasoning. Experimental results demonstrate significant performance gains, with GPT-4 improving by 8.7 points in code optimization, 13.9 points in code readability, and 21.6 points in sentiment reversal tasks, showcasing its potential to enhance the reasoning and adaptability of LLMs across various domains.

Code Prompting

Pretraining에 코드 학습시키면 Reasoning이 향상 된다고 함
근데 원인은 모름

Pre-training on code enhances the reasoning capabilities of LLMs, yet the underlying mechanisms driving this improvement remain poorly understood. To investigate this, Puerto *et al.* [2024] examines the impact of input representation on LLM reasoning, specifically exploring whether reformulating natural language (NL) problems into code can trigger conditional reasoning abilities. This led to the introduction of Code Prompting, a technique that reformulates NL tasks into structured code, enabling direct prompting of text+code LLMs without relying on external code execution. Experiments on three reasoning benchmarks, ConditionalQA, BoardgameQA, and ShARC, demonstrate that code prompts significantly outperform traditional text-based prompts. On average, GPT 3.5 achieved a performance gain of 8.42 F1 score, while Mistral showed an average improvement of 4.22 across the three datasets.

LLM에게 자연어를 제공하고 코드를 생성하게 함

Self-Harmonized Chain-of-Thought (ECHO) Prompting

While Chain-of-Thought prompting enhances reasoning in LLMs, methods like Auto-CoT, which automate demonstration generation, face challenges from misleading similarity (incorrect rationales in similar examples) and ineffective diversity (irrelevant or overly varied patterns). To address these issues, Mekala *et al.* [2024] introduced ECHO, a self-harmonized prompting framework that unifies diverse reasoning paths into a coherent pattern, balancing automation with robustness. ECHO operates through three key stages: (1) Question Clustering, where Sentence-BERT embeddings and k-means group questions into clusters; (2) Demonstration Sampling, which selects representative questions from each cluster and generates rationales using Zero-Shot-CoT; and (3) Demonstration Unification, where rationales are iteratively refined using a dynamic prompting mechanism to align reasoning patterns. This process minimizes diversity-induced noise while retaining adaptability. ECHO surpassed Auto-CoT by an average of 2.8% across 10 reasoning benchmarks (arithmetic, commonsense, symbolic) while demonstrating greater efficiency. It retained performance with 50% fewer examples, showing only a -0.8% dip compared to Few-Shot-CoT's -1.3% decline. The method also achieved 2.3% gains over Auto-CoT in Mixtral-8x7B, though it remained behind GPT-3.5, a gap attributed to differences in the quality of reasoning rationales.

AutoCoT같은 Demonstation을 자동생성하는 방식은 잘못된 근거나 유사 사례로 Diversity가 망가지고 오답을 생성할 수 있음
ECHO는 다양한 추론 경로와 일관성의 균형을 이뤄보는 방식임
1) Sentence-BERT 임베딩 + Kmeans로 Query를 클러스터링 (Query Clustering)
2) 각 클러스터의 대표 Query를 선택하고 Zeroshot CoT를 통해 합리적 추론을 생성(Demonstration Sampling)
3) 생성된 추론들을 정제하여 추론 패턴을 동적으로 일치시킴(Demonstration Unification)

LLM은 종종 중간 추론과정과 최종 결론이 불일치함 Logic-of-thought Prompting은 이를 해결하고자 함
1) 입력 텍스트로부터 명제와 논리적 관계 식별(Logic Extraction)
2) Python 기반 모듈로 대우법(contraposition) 등 형식 논리 규칙을 적용, 추가적인 논리표현 유도(Logic Extension)
3) 확장된 논리를 자연어로 변환하여 원래 프롬프트에 추가(Logic Translation)

Logic-of-thought Prompting

LLMs often exhibit unfaithful reasoning, where the generated conclusions diverge from the intermediate reasoning steps. Logic-of-Thought prompting [Liu *et al.*, 2024] is a neuro-symbolic framework developed to mitigate this issue by enriching prompts with logical information derived from propositional logic. LoT operates in three phases: (1) Logic Extraction, during which LLMs identify propositions and logical relationships from input texts; (2) Logic Extension, in which a Python-based module applies formal logical laws (e.g., contraposition) to infer additional expressions; and (3) Logic Translation, where the extended logic is rendered back into natural language and appended to the original prompt to ensure contextual fidelity. Moreover, Logic-of-thought is designed to integrate seamlessly with other prompting strategies such as CoT, Self-Consistency, and ToT prompting. Reported evaluations indicate that Logic-of-thought can improve CoT accuracy on the ReClor benchmark by 4.35%, enhance CoT prompting with Self-Consistency on LogiQA by 5%, and further boost ToT prompting performance on the ProofWriter dataset by 8%. Additionally, by preserving natural language representations throughout the process, Logic-of-Thought avoids the symbolic extraction errors that can impair other neuro-symbolic systems, such as SatLM.

대치, 대립, 대우

충실함

CoT, Self-Consistency, ToT 등 기존기법과 결합도 용이함

Instance-adaptive Prompting (IAP)

1) IAP-ss: Threshold를 충족할 때까지 Test 반복
2) IAP-mv : Majority Vote로 최종 응답 결정

Yuan *et al.* [2024] tackle the generalization constraints of static task-level prompts (e.g., "Let's think step by step") in zero-shot CoT reasoning by introducing Instance-Adaptive Prompting (IAP), a saliency-driven framework designed to dynamically tailor prompts to individual instances. Through information flow analysis of attention layers, the authors identified distinct patterns: effective reasoning correlates with strong semantic flow from questions to prompts in shallow layers and from integrated question-prompt representations to rationales in deeper layers. In contrast, fragmented or weak flows are indicative of suboptimal reasoning performance. IAP optimizes reasoning fidelity through two adaptive strategies. The first, IAP-ss (Sequential Substitution), enhances efficiency by iteratively testing prompts until predefined saliency thresholds are met. The second, IAP-mv (Majority Vote), prioritizes robustness by aggregating saliency scores across multiple prompts to determine consensus answers. Empirical evaluations underscore the broad applicability of IAP: in mathematical reasoning tasks (GSM8K, SVAMP), IAP-mv boosts the performance of LLaMA-3-8B and Qwen-14B by +1.82% and +3.31%, respectively, compared to static prompts. It achieves 19.25% accuracy on causal judgment tasks, outperforming baselines at 16.04%, and surpasses Self-Discover by +21.7% on MMLU commonsense reasoning with Qwen-14B.

End-to End DAG-Path (EEDP) Prompting

End-to-End DAG-Path (EEDP) prompting [Hong *et al.*, 2024] addresses the limitations of traditional graph-flattening methods, such as adjacency lists and edge lists, which struggle with long-distance reasoning in graph-related tasks for LLMs. EEDP's key insight is that conventional flattened representations often lose critical long-range dependencies

Graph의 주요 경로(Backbone Path)를 중심으로 정보를 구성하여 추출하는 방법

essential for effective reasoning. To mitigate this, EEDP prioritizes the main backbone paths connecting graph endpoints (nodes with zero in-degree or out-degree) while preserving adjacency lists to maintain local contextual information. The EEDP framework operates through three key stages: (1) preprocessing input graphs into directed acyclic graphs (DAGs) using breadth-first search (BFS) to eliminate cycles, (2) extracting hierarchical paths between endpoints, and (3) compressing shared path segments with a differential pointer algorithm, effectively reducing token length by 55% on molecular graphs. EEDP was evaluated on tasks such as Edge Prediction Connectivity Prediction (EPCP) and Edge Prediction Distance Prediction (EPDP) using educational (`Merged_1000`) and molecular (`ZINC_test_2500`) datasets. The evaluation results highlighted significant performance gains over traditional baselines, with EPCP showing a +10.21% accuracy improvement on `Merged_1000` and +16.76% on `ZINC_test_2500`. Similarly, EPDP achieved a +4.73% accuracy boost on `Merged_1000` and an impressive +30.13% on `ZINC_test_2500`.

LLM은 법률 등 복잡한 정보 검색에서 어려워함
추론 단계에 순차적은 제약 조건을 걸어 필터링함

Layer-of-Thoughts (LoT) Prompting

LLMs demonstrate strong performance in many reasoning tasks yet frequently face challenges with the precision–recall trade-off and explainability, particularly in complex legal retrieval scenarios. Layer-of-Thoughts (LoT) prompting [Fungwacharakorn *et al.*, 2024] introduces a hierarchical framework that leverages constraint hierarchies to structure the reasoning process, thereby enhancing both retrieval accuracy and interpretability. In the context of legal document retrieval, LoT organizes reasoning into "layer thoughts" (conceptual stages) and "option thoughts" (partial solutions), applying sequential constraints to iteratively filter and refine candidate responses. For instance, the framework employs a three-layer process: (1) a Keyword Filtering Layer (KFL) that extracts LLM-generated keywords to initially filter documents using metrics such as at-least-k; (2) a Semantic Filtering Layer (SFL) that prioritizes documents based on multi-level relevance criteria and aggregation metrics; and (3) a Final Confirmation Layer (FCL) that validates the remaining candidates against the original query. By integrating both hard constraints (required) and soft constraints (preferential), LoT not only delivers explainable reasoning but also outperforms state-of-the-art models, for example, achieving an F2 score of 0.835 (with precision of 0.838 and recall of 0.839) on Japanese Civil Law retrieval compared to 0.807 for JNLP, and reaching near-perfect recall (0.966) in German traffic law contexts.

Narrative-of-Thought (NoT) Prompting

Temporal reasoning remains a significant challenge for LLMs, particularly in inferring global temporal relationships from unordered events. To evaluate this capability, Zhang *et al.* [2024] introduced Temporal Graph Generation (TGG), a benchmark designed to assess LLMs' proficiency in constructing directed acyclic graphs (DAGs) representing event timelines. Experimental results revealed that smaller LLMs (<10B) lagged behind GPT-3.5/4 by approximately 50%, with even GPT-4 facing difficulties due to alignment constraints. To overcome these limitations, the authors proposed Narrative-of-Thought (NOT), a prompting strategy that enhances temporal reasoning without

시간 관계를 유추하는데 사용하는 프롬프팅
1) Event를 Python Class로 만들고 코드 자동완성으로 시계열 정보 명시
2) NoT 프롬프트 임시적으로 시간 관계 그래프를 생성
3) 시간관계 기반 프롬프트 제공

requiring additional model training. NOT comprises three core components: (1) Structural Representation, where events are encapsulated in a Python class and processed through code completion; (2) NOT Prompting template, which generates temporally grounded narratives to guide the construction of temporal graphs; and (3) Narrative-Aware Demonstrations, utilizing GPT-4-generated few-shot examples optimized for both conciseness and accuracy. Results demonstrated that NOT significantly improves the performance of small LLMs, with LLaMA3-8B achieving an F1 score of 42.2, closely matching GPT-3.5's 45.7, while exhibiting superior structural coherence.

Buffer of Thoughts (BoT) Prompting

다양한 과제로부터 구조화된 해결 패턴을 메타 버퍼에 저장
새로운 문제 해결 / 메타 버퍼 지속 정제 및 보완

Existing prompting methods often struggle to balance universality, efficiency, and robustness in complex reasoning. To address this, Yang *et al.* [2024] introduced Buffer of Thoughts (BoT), a framework that enhances LLMs through reusable high-level reasoning patterns. BoT overcomes the limitations of single-query methods (e.g., manual exemplar reliance) and multi-query approaches (e.g., computational inefficiency) by introducing a meta-buffer that distills "thought-templates" from diverse tasks and a dynamic buffer-manager that continuously refines them as new problems are solved. BoT retrieves task-specific thought-templates (e.g., structured problem-solving approaches) and adaptively instantiates them, mimicking human analogical reasoning to eliminate manual prompt design and recursive exploration. Experiments across 10 benchmarks demonstrate its state-of-the-art performance, achieving gains of 11% on Game of 24, 20% on Geometric Shapes, and 51% on Checkmate-in-One, while using just 12% of the computational cost of multi-query methods like Tree-of-Thoughts. Notably, BoT enhances smaller models, with Llama3-8B + BoT surpassing Llama3-70B in accuracy, showing its potential to democratize efficient reasoning at scale.

Contrastive Denoising with Noisy Chain-of-Thought (CD-CoT) Prompting

기존의 Self-consistency, Self-Correction 등은 빈도 기반 선택 등을 수행하여 논리적 오류를 잡기 어려웠음

Contrastive Denoising with Noisy Chain-of-Thought (CD-CoT) [Zhou *et al.*, 2024] addresses the challenge of "noisy rationales" in chain-of-thought prompting, where irrelevant or incorrect intermediate reasoning steps degrade LLM performance. The NoRa (Noisy Rationales) dataset highlights this issue, showing that LLMs often perform worse with flawed rationales than with no examples at all, as they tend to mimic incorrect reasoning. Existing methods like self-correction and self-consistency offer limited solutions, as self-correction fails without external feedback, and self-consistency selects frequent answers without resolving reasoning flaws. CD-CoT mitigates this by contrasting noisy rationales with clean ones, rephrasing flawed examples, selecting optimal reasoning paths, and voting on the most consistent answer. Experiments show that CD-CoT improves accuracy by 17.8% on average, significantly outperforming baselines and enhancing LLMs' robustness in reasoning-intensive tasks.

CD-CoT는 Noisy 추론과 정제된 추론의 대조학습을 통해 최적 추론 경로를 선별한 후, 가장 일관된 응답에 투표하는 방식으로 견고한 추론을 제공함

Reverse Chain-of-Thought (R-CoT) Prompting

Deng *et al.* [2024] introduced the Reverse Chain-of-Thought (R-CoT) pipeline, a novel approach to enhancing geometric reasoning in LMMs by addressing dataset limitations such as low quality, diversity, and fidelity. R-CoT operates in two

기하학 추론의 약점인 저품질 데이터셋, 낮은 다양성, 불충분한 논리 정합성 개선을 위해 사용
추론과정 생성 -> 역방향으로 문제를 생성함

stages: GeoChain, which generates high-fidelity geometric images with detailed step-by-step descriptions of geometric relationships (e.g., midlines, radii), and Reverse A&Q, which derives questions from reasoning chains using LLMs, ensuring accurate multi-step problem generation. By prioritizing answer-aware question synthesis, R-CoT mitigates visual hallucinations and reasoning errors in LMMs. The resulting GeoMM dataset includes 20 geometric shapes categorized by complexity, incorporating relational questions often missing in existing datasets like MAVIS and GeomVerse. GeoMM combines high-fidelity images with diverse Q&A pairs, enriched by geometric theorems and line operations. Experimental results demonstrate that R-CoT-trained models achieve state-of-the-art performance, with the 8B-parameter model surpassing GPT-4o by 12.5% on MathVista and 14.5% on GeoQA, while smaller models (2B, 7B) also set new benchmarks.

Chain of Draft (CoD) Prompting

CoT + Output max_length 등의 제약을 줘서 핵심만 뽑게 함

Chain of Draft (CoD) [Xu *et al.*, 2025], a novel prompting strategy designed to enhance efficiency in complex reasoning tasks. Unlike traditional CoT prompting, which emphasizes detailed step-by-step reasoning, CoD generates concise, information-dense outputs at each step, mirroring human problem-solving strategies where only essential insights are noted. While CoT improves reasoning accuracy, it often leads to verbose outputs and increased computational costs. CoD mitigates this by constraining word usage in each reasoning step, reducing latency and token consumption without sacrificing accuracy. This efficiency-oriented approach is particularly valuable for real-world applications where computational resources and response time are critical. Experimental results across arithmetic, commonsense, and symbolic reasoning benchmarks show that CoD matches or even outperforms CoT in accuracy while significantly lowering token usage and latency. In some cases, CoD achieved comparable accuracy with an 80% reduction in output tokens as well as an average latency reduction of 76.2%, demonstrating its potential as a lightweight yet effective alternative to traditional prompting strategies.

2.3 Reduce Hallucination

Retrieval Augmented Generation (RAG)

LLMs have revolutionized text generation, yet their reliance on limited, static training data hinders accurate responses, especially in tasks demanding external knowledge. Traditional prompting falls short, requiring expensive retraining. Retrieval Augmented Generation (RAG) [Lewis *et al.*, 2020] emerges as a novel solution, seamlessly weaving information retrieval into the prompting process. RAG analyzes user input, crafts a targeted query, and scours a pre-built knowledge base for relevant resources. Retrieved snippets are incorporated into the original prompt, enriching it with contextual background. The augmented prompt empowers the LLM to generate creative, factually accurate responses. RAG's agility overcomes static limitations, making it a game-changer for tasks requiring up-to-date knowledge. RAG outperformed seq2seq models and task-specific architectures on ODQA benchmarks, achieving exact match scores, reaching up to 56.8% on TriviaQA and 44.5% on Natural Questions.

샅샅이 뒤지다

훈련을 추가하지 않고, 외부 지식을 프롬프트에 추가하여 정확성을 높이는 방법
- 사용자의 입력정보를 기반으로 관련된 리소스를 찾기 위한 쿼리를 작성 -> 정보탐색
- 결과값을 기존 프롬프트에 추가하여 Context를 강화함

Reasoning과 Action을 상호보완적으로 결합하여 한번에 생성하는 방식
모델이 Task 실행 중 발생하는 Exception에 대응하며, 추론 경로와 행동계획을 유기적으로 유도/추적/갱신할 수 있게 함

ReAct Prompting

Unlike previous studies that treated reasoning and action separately, ReAct [Yao *et al.*, 2022] enables LLMs to generate reasoning traces and task-specific actions concurrently. This interleaved process enhances synergy between reasoning and action, facilitating the model in inducing, tracking, and updating action plans while handling exceptions. ReAct is applied to diverse language and decision-making tasks, showcasing its effectiveness over state-of-the-art baselines. Notably, in question answering (HotpotQA) and fact verification (Fever), ReAct addresses hallucination and error propagation issues by interacting with a simple Wikipedia API, producing more interpretable task-solving trajectories. Additionally, in interactive decision-making benchmarks like ALFWorld and WebShop, ReAct surpasses both imitation and reinforcement learning approaches, achieving notable success rates of 34% and 10%, respectively, with minimal in-context examples.

Chain-of-Verification (CoVe) Prompting

검증을 통해 할루시네이션 정제
1) 응답 생성
2) Verification 질문 계획
3) Verification 질문에 응답
4) 수정된 최종 응답 생성

To address hallucinations in LLMs, Dhuliawala *et al.* [2023] proposed Chain-of-Verification (CoVe), which involves a systematic four-step process including the model generate baseline responses, plan verification questions to check its work, answer the questions independently, and produce a revised response incorporating the verification. By verifying its work through this deliberate multi-step approach, the LLM enhances logical reasoning abilities and reduces errors even with contradictory information. CoVe emulates human verification to bolster the coherence and precision of LLM output. Experiments on list questions, QA, and long-form generation demonstrate that CoVe decreases hallucinations while maintaining facts [Sahoo *et al.*, 2024]. Focused verification questions help models identify and correct their inaccuracies.

Chain-of-Note (CoN) Prompting

Retrieval-augmented language models (RALMs) enhance large language models by incorporating external knowledge to reduce factual hallucination. However, the reliability of retrieved information is not guaranteed, leading to potentially misguided responses. Standard RALMs struggle to assess their knowledge adequacy and often fail to respond with "unknown" when lacking information. To address these challenges, Yu *et al.* [2023] introduced a novel approach to improve RALMs robustness by handling noisy, irrelevant documents and accurately addressing unknown scenarios. CoN systematically evaluates document relevance, emphasizing critical and reliable information to filter out irrelevant content, resulting in more precise and contextually relevant responses. Testing across diverse open-domain question-answering datasets demonstrated notable improvements, including a +7.9 average boost in exact match scores for noisy retrieved documents and a +10.5 enhancement in rejection rates for questions beyond pre-training knowledge.

검색된 문서의 관련성 평가를 체계적으로 수행
핵심적이고 신뢰할만한 정보를 강조하여 잡음 제거

Chain-of-Knowledge (CoK) Prompting

유효성

Traditional prompting techniques for LLMs have proven powerful in tackling basic tasks. However, their efficacy diminishes due to complex reasoning challenges, often resulting in unreliable outputs plagued by factual hallucinations and opaque thought processes. This limitation arises from their reliance

1) 사전 추론 준비 단계 : 문제 해결을 위한 Context를 정립하고 문제의 구조를 명확히 정의함
2) 동적 지식 적용 단계 : LLM 내부지식, 외부 DB, 프롬프트 내 정보 등 다양한 출처로부터 증거를 수집

on fixed knowledge sources, ineffective structured query generation, and lack of progressive correction that fails to guide the LLM adequately. Motivated by human problem-solving, CoK [Li *et al.*, 2023d] systematically breaks down intricate tasks into well-coordinated steps. The process initiates with a comprehensive reasoning preparation stage, where the context is established, and the problem is framed. Subsequently, it engages in a dynamic knowledge adaptation phase, meticulously gathering evidence from various sources, such as its internal knowledge base, external databases, and the given prompt.

2.4 User Interface

Active Prompting

Diao *et al.* [2023] introduced Active-Prompting as a solution to the challenge of adapting LLMs to diverse reasoning tasks. They address the issue by proposing Active-Prompt to enhance LLMs' performance on complex question-and-answer tasks through task-specific example prompts with chain-of-thought (CoT) reasoning. Unlike existing CoT methods that rely on fixed sets of human-annotated exemplars, Active-Prompt introduces a mechanism for determining the most impactful questions for annotation. Drawing inspiration from uncertainty-based active learning, the method utilizes various metrics to characterize uncertainty and selects the most uncertain questions for annotation. Active-Prompting exhibits superior performance, outperforming self-consistency by an average of 7.0% and 1.8% across eight complex reasoning tasks in text-davinci-002 and code-davinci-002, respectively, showcasing state-of-the-art results.

2.5 Fine-Tuning and Optimization

Automatic Prompt Engineer (APE)

While crafting effective prompts for LLMs has traditionally been a laborious task for expert annotators, Zhou *et al.* [2022] introduced Automatic Prompt Engineer (APE) as an innovative approach to automatic instruction generation and selection for LLMs. APE sheds the limitations of static, hand-designed prompts by dynamically generating and selecting the most impactful prompts for specific tasks. This ingenious method analyzes user input, crafts candidate instructions, and then leverages reinforcement learning to choose the optimal prompt, adapting it on the fly to different contexts. Extensive tests on the diverse BIG-Bench suite and the CoT reasoning task revealed APE's prowess, exceeding human-authored prompts in most cases (19 out of 24 tasks) and significantly boosting LLMs reasoning abilities. This breakthrough in automatic prompt engineering paves the way for LLMs to tackle a wider range of tasks with greater efficiency and adaptability, unlocking their full potential across diverse applications.

2.6 Knowledge-Based Reasoning and Generation

Automatic Reasoning and Tool-use (ART)

The limited reasoning abilities and lack of external tool utilization hinder the potential of LLMs in complex tasks. Paranjape *et al.* [2023] introduced Automatic Reasoning and Tool-use (ART) to tackle this critical barrier that empowers LLMs to reason through multi-step processes and seamlessly integrate external expertise. ART bridges the reasoning gap, enabling LLMs to tackle complex problems and expand beyond simple text generation. By integrating external tools for specialized knowledge and computations, ART unlocks unprecedented versatility and informs LLM outputs with real-world relevance. This allows LLMs to contribute to diverse fields like scientific research, data analysis, and even decision-making support. Moving beyond traditional prompting techniques, ART automates reasoning steps through structured programs, eliminating the need for laborious hand-crafting. Its dynamic tool integration ensures smooth collaboration, pausing generation to incorporate external tool outputs and seamlessly resuming the flow. Empirical evidence on challenging benchmarks (BigBench and MMLU) demonstrates ART's effectiveness, surpassing traditional prompting and even matching hand-crafted demonstrations in some cases.

2.7 Improving Consistency and Coherence

Contrastive Chain-of-Thought (CCoT) Prompting

Traditional CoT prompting for LLMs often misses a crucial element: learning from mistakes. That is where Contrastive Chain-of-Thought Prompting (CCoT) [Chia *et al.*, 2023] dives in, providing both valid and invalid reasoning demonstrations alongside original prompts. Imagine exploring a map with the right path and the wrong turns to avoid – that is the advantage of contrastive CoT! This dual-perspective approach, tested on reasoning benchmarks like SQuAD and COPA, pushes LLMs to step-by-step reasoning, leading to 4-16% improvements in strategic and mathematical reasoning evaluations compared to traditional CoT, further improved by approximately 5% when integrated with self-consistency techniques. However, questions remain about this technique, such as the automated generation of contrasting demonstrations for diverse problems and its applicability to other NLP tasks beyond reasoning.

2.8 Managing Emotions and Tone

Emotion Prompting

While LLMs demonstrate impressive capabilities on various tasks, their ability to comprehend psychological and emotional cues remains uncertain. The study by Li *et al.* [2023a] addressed the uncertainty surrounding LLMs' ability to comprehend emotional cues by introducing EmotionPrompt. Drawing inspiration from psychological research on language's impact on human performance, they append 11 emotional stimulus sentences to prompts to enhance LLM emotional intelligence. Experimental results demonstrate seamless integration of these stimuli, significantly improving LLM performance across various tasks. EmotionPrompt demonstrates an 8.00% relative improvement in instruction induction and an impressive 115% boost in BIG-Bench tasks, underscoring its efficacy in augmenting LLM capabilities in processing affective signals. An evaluation involving 106 participants reveals an average improvement of 10.9% in performance, truthfulness, and responsibility metrics for generative tasks when employing EmotionPrompt compared to standard prompts.

2.9 Code Generation and Execution

Scratchpad Prompting

Despite the prowess of Transformer-based language models in generating code for basic programming tasks, they encounter

모델이 바로 정답을 출력하는 대신, 중간 과정을 자유롭게 표현할 수 있는 작업공간(ScratchPad)을 먼저 작성하게 한 뒤, 최종 정답을 출력하도록 함 (메모장에 계산 과정 적는거랑 유사하게)

challenges in complex, multi-step algorithmic calculations requiring precise reasoning. Addressing this, Nye *et al.* [2021] introduce a novel approach, centered on task design rather than model modification, introduce a 'scratchpad' concept. The proposal enables the model to generate an arbitrary sequence of intermediate tokens before providing the final answer. Scratchpad Prompting technique outperforms (Mostly Basic Python Programming) MBPP-aug with a 46.8% success rate. Combining CodeNet and single-line datasets yields the highest performance, achieving 26.6% correct final outputs and 24.6% perfect traces. Scratchpad prompting technique faces limitations, including a fixed context window size of 512 tokens and a dependency on supervised learning for scratchpad utilization.

Program of Thoughts (PoT) Prompting

Numerical Reasoning을 위함

Codex등을 통해 Reasoning을 Python 프로그램으로 표현하여 활용

Language models are suboptimal for solving mathematical expressions due to their proneness to arithmetic errors, incapability in handling complex equations, and inefficiency in expressing extensive iterations. To enhance numerical reasoning in language models, Chen *et al.* [2022] presents Program-of-Thoughts (PoT) prompting, advocating the use of external language interpreters for computation steps. PoT enables models like Codex to express reasoning through executable Python programs, resulting in an average performance improvement of approximately 12% compared to CoT prompting on datasets involving mathematical word problems and financial questions.

Structured Chain-of-Thought (SCoT) Prompting

CoT가 자연어는 효과적이지만 코드 생성을 잘 못함

코드 생성에 필수적인 Sequence, Branch, Loop 등을 추론 단계에 명시함

LLMs have exhibited impressive proficiency in code generation. The widely used CoT prompting involves producing intermediate natural language reasoning steps before generating code. Despite its efficacy in natural language generation, CoT prompting demonstrates lower accuracy when applied to code generation tasks. Li *et al.* [2023c] introduce Structured Chain-of-Thought (SCoTs) as an innovative prompting technique tailored specifically for code generation. By incorporating program structures (sequence, branch, and loop structures) into reasoning steps, SCoT prompting enhances LLMs' performance in generating structured source code. This approach explicitly guides LLMs to consider requirements from the source code perspective, improving their overall effectiveness in code generation compared to CoT prompting. The authors validated the effectiveness of SCoT on ChatGPT and Codex across three benchmarks (HumanEval, MBPP, and MBCPP) and demonstrated a superior performance over the CoT prompting by up to 13.79%.

Chain-of-Code (CoC) Prompting

While CoT prompting has proven very effective for enhancing Language models (LMs) semantic reasoning skills, it struggles to handle questions requiring numeric or symbolic reasoning. Li *et al.* [2023b] introduce Chain-of-Code (CoC) as an extension to improve LM reasoning by leveraging codewriting for both logic and semantic tasks. CoC encourages LMs to format semantic sub-tasks as flexible pseudocode, allowing an interpreter to catch undefined behaviors and simulate them with an "LMulator." Experiments demonstrate CoC's superiority over Chain of Thought and other baselines, achieving an 84% accuracy on BIG-Bench Hard, a 12% gain. CoC proves effective with both large and small models, expanding LMs' ability to correctly answer reasoning questions by incorporating a "think in code" approach.

CoT가 수치나 기호 기반 문제에 한계가 있음
자연어 대신 의사코드 형태로 논리적 / 의미적 하위 과제를 표현하도록 함

2.10 Optimization and Efficiency

자연어 프롬프트를 이용해 모델이 반복적으로 해답을 생성하여 스스로 최적해에 수렴하도록 유도

Optimization by Prompting (OPRO)

In various domains, optimization is a fundamental process often involving iterative techniques. Yang *et al.* [2023] introduce Optimization by PROmpting (OPRO), a novel approach that leverages LLMs as optimizers. Unlike traditional methods, OPRO utilizes natural language prompts to iteratively generate solutions based on the problem description, enabling quick adaptation to different tasks and customization of the optimization process. The potential of LLMs for optimization is demonstrated through case studies on classic problems like linear regression and the traveling salesman problem. Additionally, it explores the optimization of prompts to maximize accuracy in natural language processing tasks, highlighting the sensitivity of LLMs. The experiments show that optimizing prompts for accuracy on a small training set effectively translates to high performance on the test set. OPRO leads to a significant performance boost, with the most effective prompts optimized by OPRO outperforming human-designed prompts by up to 8% on the GSM8K dataset and up to 50% on challenging tasks in Big-Bench.

2.11 Understanding User Intent

먼저 질문을 재구성하고, 응답을 생성하도록 유도함

Rephrase and Respond (RaR) Prompting

The study by Deng *et al.* [2023] brings attention to an often-neglected dimension in exploring LLMs: the disparity between human thought frames and those of LLMs and introduces Rephrase and Respond (RaR). RaR allows LLMs to rephrase and expand questions in a single prompt, demonstrating improved comprehension and response accuracy. The two-step RaR variant, incorporating rephrasing and response LLMs, achieves substantial performance enhancements across various tasks. The study highlights that in contrast to casually posed human queries, the rephrased questions contribute to enhanced semantic clarity and the resolution of inherent ambiguity. These findings offer valuable insights for understanding and enhancing the efficacy of LLMs across various applications.

2.12 Metacognition and Self-Reflection

Take a Step Back Prompting

Addressing the persistent challenge of complex multi-step reasoning, Zheng *et al.* [2023] introduced the Step-Back prompting technique, tailored explicitly for advanced language models like PaLM-2L. This innovative approach empowers models to engage in abstraction, extracting high-level concepts and fundamental principles from specific instances. The Step-Back prompting method involves a two-step procedure, integrating Abstraction and Reasoning. Through extensive experiments, applying Step-Back Prompting to PaLM-2L in diverse reasoning-intensive tasks such as STEM, Knowledge QA, and Multi-Hop Reasoning, the results demonstrate a substantial enhancement in reasoning capabilities. Noteworthy performance boosts are observed, with improvements in tasks like MMLU Physics and Chemistry by 7%, TimeQA by 27%, and MuSiQue by 7%.

Multi Step Reasoning을 위한 기법, 2단계로 나뉜다
모델이 Reasoning : 고차 개념을 바탕으로더 높은 수준의 개념과 근본 원리를 추출하도록 유도함
1) Integrating Abstraction : 주어진 문제에서 핵심 개념이나 일반 원칙 도출
2) Reasoning : 고차 개념을 바탕으로 논리적 추론 전개

Table 1: Summary of prevalent prompting techniques of LLMs based on the following factors: application, prompt acquisition, prompt turn, language model, dataset, and metrics.

Application	Prompting Technique	Comparison Scope				
		Prompt Acquisition	Prompt Turn	Language Model(s)	Dataset	Metric(s)
New Tasks Without Training Data	Zero-shot	Manual	Single	GPT-2	Arithmetic,Symbolic	Accuracy, ROUGE Score
	Few-shot	Manual	Single	GPT-3	NaturalQS, WebQS, TriviaQA	Accuracy
Reasoning and Logic	CoT	Manual	Multi	PaLM 540B	GSM8K	Accuracy
	LogiCoT	Manual	Multi	Vicuna-33b, GPT-4	GSM8K, AQuA, SocialQA	Accuracy
	CoS	Manual	Multi	`gpt-3.5-turbo`, GPT-4	SPARTUN	Accuracy, Precision, Recall
	Auto-CoT	LM Generated	Multi	GPT-3	Arithmetic, Symbolic	Accuracy
	Self-Consistency	Manual	Single	PaLM 540B	Arithmetic, Commonsense	Accuracy
	ToT	Retrieval Based	Multi	GPT-4	Game of 24, Creative Writing	Success Rate
	GoT	Retrieval Based	Multi	T5-large	GSM8K, ScienceQA	ROUGE Score
	S2A	Manual	Single	Llama 2-70B	QA,GSM8K	Accuracy
	ThoT	Hybrid	Multi	`gpt-3.5-turbo`, Llama 2-70b-chat	PopQA, EntityQ, MTCR	Exact Match (EM) Score
	Chain of Table	Manual	Multi	GPT 3.5, LLaMA 2	TabFact, WikiTQ	BLEU, ROUGE Score
	Self-Refine	Manual	Multi	GPT-3.5,GPT-4	7 diverse tasks(e.g., Dialogue Response,Math Reasoning)	Task-specific (Accuracy, Human Preference)
	Code Prompting	LM Generated	Multi	GPT 3.5, Mixtral	CondQA,ShaRC,BGQA	F1
	ECHO	Hybrid	Multi	`gpt-3.5-Turbo-0301`	Arithmetic,Commonsense,Symbolic	Accuracy
	Logic-of-thought	LM Generated	Multi	GPT 3.5-turbo, GPT-4	ReClor, LogiQA, RuleTaker, ProofWriter, FOLIO	Accuracy
	IAP	Manual	Multi	LLaMA-3-8B-Instruct, Qwen-14B-Chat	Math,Logic,Commonsense	Accuracy
	EEDP	Manual	Single	GPT-4-turbo	Merged 1000, ZINC test 2500	Accuracy
	LoT	LM Generated	Multi	GPT-4o	Japanese Civil Law,Normative sentence	Precision,Recall,F2
	NoT	LM Generated	Single	GPT-3.5,GPT-4, Mistral-7B,LLaMA3-8B	ProScript,Schema-11,WikiHow Script	F1,GED
	BoT	LM Generated	Multi	Llama3-8B, Llama3-70B	10 reasoning-intensive tasks (e.g., Game of 24, Geometric Shapes)	Accuracy
	CD-CoT	Manual	Single	`gpt-3.5-turbo-0613`, Gemini-Pro(and others)	Multiple tasks (e.g. BIG-Bench subsets, commonsense QA, etc.)	Accuracy, Solve Rate, Human Preference
	R-CoT	Manual	Single	GPT4o, R-CoT-8B(and others)	GeoMM,MathVista,GeoQA	Accuracy
	CoD	Hybrid	Single	GPT-4o,Claude 3.5 Sonnet	Arithmetic, Commonsense, Symbolic	Accuracy
Reduce Hallucination	CoVe	Retrieval Based	Multi	Llama 65B	Wikidata, QUEST, MultiSpanQA	Precision, F1
	ReAct	Retrieval Based	Multi	PaLM-540B, GPT-3	HotpotQA, FEVER	Exact Match (EM), Accuracy
	RAG	Retrieval Based	Single	RAG-Token, RAG-Seq.	MSMARCO, SearchQA	ROUGE, BLEU score
	CoN	LM Generated	Multi	Llama 2, DPR	NQ, TriviaQA, WebQ	Exact Match (EM), F1 Score
	CoK	LM Generated	Multi	`gpt-3.5-turbo-0613`	HotpotQA, FEVER, MedMCQA, MMLU Physics and Biology	Exact Match (EM), Accuracy
User Interaction	Active-Prompt	Manual	Single	`code-davinci-002`, `text-davinci-003`	Arithmetic, Commonsense, Symbolic	Disagreement, Entropy Variance, Self-confidence Score
Fine-Tuning and Optimization	APE	LM Generated	Single	`text-curie-001`, `text-davanci-002`	BBII, TruthfulQA	Execution accuracy, Log probability, Efficient score estimation
Knowledge-Based Reasoning and Generation	ART	Hybrid	Multi	GPT-3 (175B)	BigBench, MMLU	Accuracy
Improving Consistency and Coherence	CCoT	LM Generated	Multi	`gpt-3.5-turbo-0301`	Arithmetic, Factual QA	Accuracy
Managing Emotions and Tone	Emotion Prompting	Manual	Single	GPT-4	BIG-Bench, Instruction Induction	Accuracy
Code Generation and Execution	SCoT	Hybrid	Multi	ChatGPT, Codex	HumanEval, MBPP, MBCPP	pass@k
	PoT	Manual	Single	`gpt-3.5-turbo`	GSM8K, SVAMP, FinQA	Exact Match(EM) Score
	CoC	Manual	Single	`text-davinci-003`, `gpt-3.5-turbo`	BIG-Bench Hard	Accuracy
	Scratchpad Prompting	Manual	Single	GPT-3	MBPP, MBPP-aug	Accuracy
Optimization and Efficiency	OPRO	Manual	Single	PaLM 2-L-IT, text-bison	GSM8K, BIG-Bench Hard	Accuracy
Understanding User Intent	RaR	Manual	Single	`GPT-4-0613`	Knowledge, Symbolic	Accuray, Fair Score, Language Modeling Score
Metacognition and Self-Reflection	Take a Step Back	Manual	Single	PaLM2-L, GPT-4	MMLU-Physics, MMLU-Chemistry TimeQA, SituatedQA, StrategyQA	Accuracy

3 Conclusion

In the domain of artificial intelligence, prompt engineering has become a transformative force, unlocking the vast potential of LLMs. This survey paper aims to serve as a foundational resource that systematically categorizes 41 distinct prompt engineering techniques based on their targeted functionalities, inspiring further research and empowering innovators in the evolving landscape of prompt engineering. The analysis spans applications, models, and datasets, shedding light on the strengths and limitations of each approach. Furthermore, we have added a diagram and a table to highlight the important points. Despite the remarkable successes, challenges persist, including biases, factual inaccuracies, and interpretability gaps, necessitating further investigation and mitigation strategies. The future of prompt engineering holds immense potential, with emerging trends like meta-learning and hybrid prompting architectures promising amplified capabilities. However, ethical considerations are paramount, emphasizing responsible development and deployment to ensure positive integration into our lives.

References

Hyojin Bahng, Ali Jahanian, Swami Sankaranarayanan, and Phillip Isola. Exploring visual prompts for adapting large-scale models. *arXiv preprint arXiv:2203.17274*, 2022.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022.

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.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*, 2023.

Yihe Deng, Weitong Zhang, Zixiang Chen, and Quanquan Gu. Rephrase and respond: Let large language models ask better questions for themselves. *arXiv preprint arXiv:2311.04205*, 2023.

Linger Deng, Yuliang Liu, Bohan Li, Dongliang Luo, Liang Wu, Chengquan Zhang, Pengyuan Lyu, Ziyang Zhang, Gang Zhang, Errui Ding, Yingying Zhu, and Xiang Bai.

R-cot: Reverse chain-of-thought problem generation for geometric reasoning in large multimodal models, 2024.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. *arXiv preprint arXiv:2309.11495*, 2023.

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

Wachara Fungwacharakorn, Nguyen Ha Thanh, May Myo Zin, and Ken Satoh. Layer-of-thoughts prompting (lot): Leveraging llm-based retrieval with constraint hierarchies, 2024.

Bin Hong, Jinze Wu, Jiayu Liu, Liang Ding, Jing Sha, Kai Zhang, Shijin Wang, and Zhenya Huang. End-to-end graph flattening method for large language models, 2024.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Yun-Ze Song, Wai Lam, and Yue Zhang. Chain-of-symbol prompting elicits planning in large langauge models, 2023.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Cheng Li, Jindong Wang, Yixuan Zhang, Kaijie Zhu, Wenxin Hou, Jianxun Lian, Fang Luo, Qiang Yang, and Xing Xie. Large language models understand and can be enhanced by emotional stimuli. *arXiv preprint arXiv:2307.11760*, 2023.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. *arXiv preprint arXiv:2312.04474*, 2023.

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. Structured chain-of-thought prompting for code generation. *arXiv preprint arXiv:2305.06599*, 2023.

Xingxuan Li, Ruochen Zhao, Yew Ken Chia, Bosheng Ding, Shafiq Joty, Soujanya Poria, and Lidong Bing. Chain-of-knowledge: Grounding large language models via dynamic knowledge adapting over heterogeneous sources, 2023.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Xingyu Wang, Jiaxing Wang, Hailong Yang, and Jing Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models, 2024.

Jieyi Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback, 2023.

Rajasekhar Reddy Mekala, Yasaman Razeghi, and Sameer Singh. Echoprompt: Instructing the model to rephrase queries for improved in-context learning, 2024.

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

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*, 2023.

Haritz Puerto, Martin Tutek, Somak Aditya, Xiaodan Zhu, and Iryna Gurevych. Code prompting elicits conditional reasoning abilities in text+code llms, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Pranab Sahoo, Prabhash Meharia, Akash Ghosh, Sriparna Saha, Vinija Jain, and Aman Chadha. A comprehensive survey of hallucination in large language, image, video and audio foundation models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 11709–11724, 2024.

SM Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 2024.

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

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. Chain-of-table: Evolving tables in the reasoning chain for table understanding, 2024.

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

Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models, 2023.

Silei Xu, Wenhao Xie, Lingxiao Zhao, and Pengcheng He. Chain of draft: Thinking faster by writing less, 2025.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Ling Yang, Zhaochen Yu, Tianjun Zhang, Shiyi Cao, Minkai Xu, Wentao Zhang, Joseph E Gonzalez, and Bin Cui. Buffer of thoughts: Thought-augmented reasoning with large language models. *Advances in Neural Information Processing Systems*, 2024.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Yao Yao, Zuchao Li, and Hai Zhao. Beyond chain-of-thought, effective graph-of-thought reasoning in large language models. *arXiv preprint arXiv:2305.16582*, 2023.

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. Chain-of-note: Enhancing robustness in retrieval-augmented language models, 2023.

Xiaosong Yuan, Chen Shen, Shaotian Yan, Xiaofeng Zhang, Liang Xie, Wenxiao Wang, Renchu Guan, Ying Wang, and Jieping Ye. Instance-adaptive zero-shot chain-of-thought prompting, 2024.

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

Xinliang Frederick Zhang, Nick Beauchamp, and Lu Wang. Narrative-of-thought: Improving temporal reasoning of large language models via recounted narratives, 2024.

Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter. Enhancing zero-shot chain-of-thought reasoning in large language models through logic, 2023.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H Chi, Quoc V Le, and Denny Zhou. Take a step back: evoking reasoning via abstraction in large language models. *arXiv preprint arXiv:2310.06117*, 2023.

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.

Yucheng Zhou, Xiubo Geng, Tao Shen, Chongyang Tao, Guodong Long, Jian-Guang Lou, and Jianbing Shen. Thread of thought unraveling chaotic contexts. *arXiv preprint arXiv:2311.08734*, 2023.

Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 123846–123910. Curran Associates, Inc., 2024.