Software Engineering 2024, 11(2): 19-24
DOI: 10.5923/j.se.20241102.02

Prompt Engineering for Generative AI: Practical Techniques and Applications

Syed Aamir Aarfi[1,*], Nashrah Ahmed[2]

[1]IT Senior Product Manager, Amazon.com Inc., Seattle WA, USA
[2]Computer Science Engineer, Integral University, Lucknow UP, India

Abstract Large language models (LLMs) have revolutionized the field of natural language processing, demonstrating remarkable capabilities in tasks such as text generation, summarization, and question-answering. However, their performance is highly dependent on the quality of the prompts or inputs provided. Prompt engineering, the practice of crafting prompts to guide LLMs towards desired outputs, has emerged as a critical area of study and application. This paper provides an analysis of various prompt engineering techniques, ranging from basic methods to advanced strategies, aimed at enhancing the performance and reliability of generative AI systems. Through illustrative examples and theoretical insights, we demonstrate how these techniques can unlock the full potential of LLMs across diverse real-world use cases. Our findings contribute to the growing body of knowledge in this rapidly evolving field and offer practical guidance for Information (IT) professionals and researchers working with generative AI applications.

Keywords Generative AI, Prompt engineering, Chain of thoughts, Large language models (LLMs), RAG

1. Introduction

The rise of large language models (LLMs) has revolutionized the field of natural language processing (NLP) and artificial intelligence (AI). These powerful models, pretrained on vast amounts of data, have exhibited remarkable capabilities in various tasks, from text generation and summarization to question-answering and language understanding. However, the performance of LLMs is highly contingent on the prompts or inputs provided to them. Prompt engineering, the practice of constructing prompts in specific ways to guide LLMs towards desired outputs, has emerged as a crucial area of study and application. This paper aims to provide a comprehensive overview of prompt engineering techniques, both basic and advanced, to enable IT professionals and researchers to leverage the full potential of generative AI systems effectively. We explore methods such as few-shot prompting, chain-of-thought (CoT) prompting, self-consistency, retrieval-augmented generation (RAG), and automated prompt engineering (APE). Through a combination of theoretical analysis and illustrative examples, we demonstrate how these techniques can enhance the performance and reliability of LLM deployments across various real-world use cases.

* Corresponding author:
aamiraarfi@gmail.com (Syed Aamir Aarfi)
Received: Sep. 19, 2024; Accepted: Oct. 3, 2024; Published: Nov. 19, 2024
Published online at http://journal.sapub.org/se

2. Foundation Models and Instructions

Tuned LLMs Foundation models, such as GPT-3, PaLM, and others, are large language models pretrained on vast amounts of unlabelled text data. This pretraining allows them to acquire broad knowledge and capabilities that can be adapted to various downstream tasks through prompting, a process known as transfer learning. The ability to rapidly adapt to new tasks with minimal task-specific fine-tuning is a key advantage of foundation models, making them highly versatile and efficient.

While foundation models acquire broad knowledge during pretraining, they may not necessarily excel at following instructions or prompts out of the box. Instruction-tuned LLMs, such as InstructGPT, Claude, and others, are specifically trained to interpret and execute text instructions accurately. These models are fine-tuned on datasets of instructions and corresponding outputs, enabling them to generate content that closely aligns with user intent and follows provided guidelines.

3. Prompts in LLM

Figure 1. Prompt input and output to LLMs

Prompts are instructions given to the LLMs to generate a response for a specific task. We can elaborate the task set conditions, show examples, or even define the output format of the expected response from the model.

A prompt may have following parts (Table 1):

Instructions: Task or instructions for the LLM model (e.g., summarise, give examples, calculate, etc).

Context: Other relevant information to augment the model response. (e.g., use this structure, recent news, etc)

Output structure: Type or format of the expected output generated by the LLM. (e.g., bullet points, json file, etc)

Input: Input query or question to get responses from the LLM model.

Table 1. Parts of an LLM prompt – An example

Context	Below is the recent customer chat history.
Instruction	Summarize the pain point of the customer
Output Structure	in one line.
Input	Recent chat history of the customer Agent: "Hi there, how can we help?" Customer: "I can't buy ABC book" Customer: "The buy button isn't working" Agent: "This book is out of stock"

Different LLM models may require prompts to be provided in a specific format or structure. Anthropic's Claude LLM model is trained on human or assistant way to interaction, and this needs to be incorporated in the prompt. Claude e.g., Human: What is the sky blue? While, Open Assistant LLM model requires specific tokens to be configured as parts of the prompt. Open Assistant e.g., <*lprefix_beginl*> You are a large language model that wants to be helpful. <*lprefix_endl*> <*lprompterl*> Why is the sky blue*?* <*lendoftextl*> <*lassistantl*>.

To generate meaningful responses from LLMs, it is important to write clear, and specific instructions. It is preferable to highlight the part of the prompt that model should consider, add details, and define constraints to force the model generate responses that can be logical and useful. This will require an iterative approach, and if the model requires multiple steps, we may instruct the model to follow a step-by-step approach. Depending upon the complexity of the task, the need for prompts engineering technique implementation may vary from basic to advance.

4. Basic Prompt Engineering

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks through prompting techniques that guide the model's generation. The most basic approach is zero-shot prompting, where the LLM attempts a task solely based on the prompt describing it, leveraging the broad knowledge acquired during pretraining. However, providing examples in one-shot or few-shot prompts can further ground the task and steer the LLM toward the desired output structure. A step further in prompting strategy is chain-of-thought (CoT) prompting, which encourages the LLM to generate step-by-step reasoning before producing the final answer, potentially improving performance on tasks requiring logical reasoning. Through examples, we will learn how these basic LLM prompt techniques can be used in different use cases.

4.1. Zero-Shot Prompt

Zero-shot prompting is the simplest prompting strategy, where the model is given only the task description or input, without any additional examples or guidance. This approach, as shown in Table 2, tests the broad capabilities acquired by the LLM during pretraining.

Table 2. Zero-shot LLM Prompt

Prompt	Summarize the key points of the article: "The Future of Robotics and Its Impact on society… [article text]".
LLM Response	[LLM generates a summary based solely on its pretraining knowledge, without any additional context or examples.]

4.2. One-Shot and Few-Shot Prompt

One-shot prompting involves providing a single example before the test input (Table 3), while few-shot prompting provides multiple examples (Table 4). These examples demonstrate the desired output format or structure, grounding the task for the model.

Table 3. One-shot LLM Prompt

Prompt	TRUE or FALSE: Aurora lights see often seen in Northern hemisphere? The answer is TRUE TRUE or FALSE: Does it snow in near equator?
LLM Response	The answer is FALSE

One-shot is suitable for simple task using a capable LLM model where one example might be enough. However, few-shot prompts can be used for complex tasks, and less capable models. Examples only guide the model but do not generally enhance it. Sometimes with capable models, single-shot prompts may yield effective responses.

Table 4. Few-shot LLM Prompt

Prompt	Lamb :: Sheep: Kid :: ? Answer is Goat Baby :: Human: Calf :: ? Answer is Whale Puppy :: Dog : Cub :: ?
LLM Response	Answer is Lion

4.3. Chain-of-Thought Prompt

Wei et al (2022) [1] demonstrated that Chain-of-thought (CoT) prompting technique encourages LLMs to generate a step-by-step reasoning process, explaining their thought process before producing the final answer. This can be facilitated through few-shot CoT prompting or zero-shot CoT prompting with instructions like "Let's think step by step.". Show examples to LLMs with reasoning, so that the LLM can generate reasoned response. This reasoning of response improves the accuracy of answer as compared to without reasoning.

While Few-shot CoT can improve results, Kojima et al (2022) [3] discovered that appending "Let's think step by step" instructions to prompt elicits reasoning chain by the LLM which generates more accurate responses. LLMs models when forced to reason, they tend to generate more precise answer by using zero shot prompts techniques as demonstrated by Kojima et al (2022) [3] in Table 6.

CoT can improve the performance of the LLMs on tasks where reasoning is required such as common sense, symbolic or arithmetic. These prompts can be designed for specific types of problems. As per Wei et al (2022), the performance gains are visible in LLM models with over 100 billion parameters. Smaller models may generate illogical CoT based responses, with a lower precision.

5. Advance Prompt Engineering

All As large language models continue to advance, effective techniques for tuning and optimizing their parameters have become increasingly crucial. Popular approaches like self-consistency prompting and tree-of-thoughts prompting leverage diverse reasoning paths and systematic exploration to enhance model performance on tasks requiring logical reasoning and problem-solving. However, embedding strategies using tools like Langchain also play a vital role, allowing models to leverage external knowledge sources through retrieval-augmented generation. Determining when and how to apply model parameter tuning is a key consideration, as it can significantly improve output quality and consistency, particularly for complex or specialized tasks. This has given rise to automated prompt engineering methods that aim to streamline and optimize the prompting process itself, further unlocking the potential of these powerful language models.

5.1. Self-Consistency Prompt

Self-consistency builds on chain-of-thought (CoT) prompting by generating multiple, diverse reasoning paths (Figure 2) through few-shot CoT prompting and using them to verify the consistency of the responses. Wang et al. (2022) [3] demonstrated that self-consistency increases the accuracy of CoT reasoning in LLMs. This approach aims to boost performance on arithmetic and common-sense reasoning tasks by exploring multiple CoTs of reasonings, and then deriving a final answer through voting as given in Table 7.

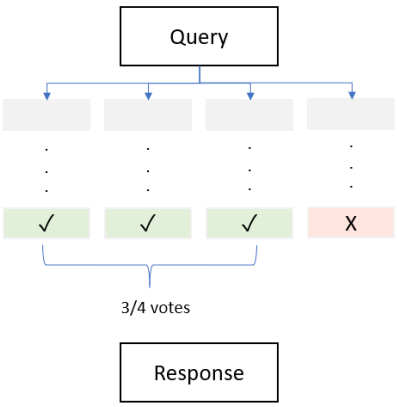

Figure 2. Self-Consistency Prompt voting technique

Table 5. Few-shot vs Few CoT LLM Prompt

	Few-shot	Few-shot CoT
Prompt	**Question:** Sam has 10 snicker bars. He buys 3 packs of snicker bars. Each pack contains 6 snicker bars. How many snicker bars does same have now? **Answer:** The answer is 28. **Question:** Restaurant has 12 waiters. If 3 of them left, and they hired 6 more. How many waiters are left?	**Question:** Sam has 10 snicker bars. He buys 3 packs of snicker bars. Each pack contains 6 snicker bars. How many snicker bars does same have now? **Answer:** Sam started with 5 snicker bars. 3 packs of 6 snickers bars are 3*6 = 18 snicker bars, i.e., adding 10+18 = 28 snicker bars. The answer is 28. **Question:** Restaurant has 12 waiters. If 3 of them left, and they hired 6 more. How many waiters are left?
LLM Response	**Answer:** The answer is 27. (**X** *Wrong answer*)	**Answer:** The restaurant has 12 waiters. 3 of them left, 12 – 3 = 9 waiters. They hired 6 more, now adding 9 + 6 = 15 waiters. The answer is 15. (✓ *Right answer*)

Table 6. Zero-shot vs Zero-shot CoT: Kojima et al (2022) LLM Prompt instruction with reasoning "Let's think step by step"

	Zero-shot	Zero-shot CoT
Prompt	**Question:** A juggler can juggle 16 balls. Half of the balls are gold balls and half of the golf balls are blue. How many blue golf balls are there? **Answer:** The answer(Arabic numeral) is	**Question:** A juggler can juggle 16 balls. Half of the balls are gold balls and half of the golf balls are blue. How many blue golf balls are there? **Answer:** Let's think step by step.
LLM Response	**Answer:** 8 (**X** *Wrong answer*)	**Answer:** There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls. (✓ *Right answer*)

Table 7. Self-consistency Prompt using diverse reasoning CoT

Prompt	Diverse Reasoning(s)	Possible Answers(s)	LLM Response
Question: If there are 3 oranges in a basket, and 3 more are added. How many oranges are there in the basket? **Answer:** There are 3 oranges in the basket. 2 more is added. Now we 3 + 2 = 5 oranges. The answer is 5. **Question:** Sam's sheeps deliver 10 lambs a month. He keeps 4 lambs and sells the remaining lambs for $30 each. How much does he earn every month?	He sells 10 - 4 = 6 lambs He earns $30*6 = $180	The answer is $180 (✓ *Right answer*)	The answer is $180 (✓ 3/4 – Majority *voted answer*)
	This means that Sam sells the remaining lambs for $30* (10-4) = $180	The answer is $180 (✓ *Right answer*)	
	He keeps 4 lambs. He then he sells remaining 10-4 = 6 lambs. So, he earns $30*6 = $180 every month	The answer is $180 (✓ *Right answer*)	
	He has 10 + 4 -6 = 8 lambs He earns $30*8 = $240	The answer is $240 (**X** *Wrong answer*)	

Table 8. Tree of thought (ToT) prompt – An Example

Prompt	Generate three short and crisp copies for an advertisement promoting a new eco-friendly product line. Thought 1: Highlight the product's environmental benefits and sustainability features. Thought 2: Appeal to consumers' desire to make a positive impact on the planet. Thought 3: Emphasize the high quality and durability of the products. Thought 4: Contrast the eco-friendly line with conventional, wasteful alternatives.
LLM Response	[LLM expands on the thought tree, evaluating and refining the potential ad copy ideas.]

Self-consistency is compute intensive as compared to CoT. Practically, it elicits 5-10 smaller number of reasoning paths, and often responses saturate. Another drawback of self-consistency is that if the CoT itself is not eliciting error prone reasoning, then self-consistency will not show any improvement. Further model tuning and training will be a better option in that case.

5.2. Tree-of-Thoughts Prompt

Tree-of-thoughts (ToT) prompting is a strategy that guides LLMs to generate, evaluate, expand on, and decide among multiple potential solutions. Similar to how humans approach problem-solving, ToT encourages the exploration of "thoughts" that serve as intermediate steps toward solving a problem. The LLM generates these thoughts via chain-of-thought prompting, and a tree-branching technique (e.g., breadth-first or depth-first search) is used to systematically explore and evaluate the potential solutions, enabling look-ahead and backtracking as given in Figure 3.

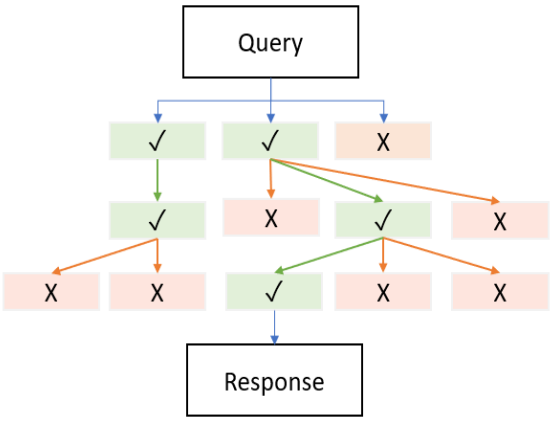

Figure 3. Tree of thoughts (ToT) Prompt tree branch technique

Tao et al (2023) [4], and Long et al (2023) [5] research indicate that Tree-of-thoughts elicits explorations over "thoughts" which paves way for general problem solving via LLMs. In table 8, we added four thoughts as an input to the prompt, and it will elicit more human like problem solving technique to generate responses.

5.3. Retrieval-Augmented Generation (RAG) Prompt

While LLMs possess remarkable knowledge acquired during pretraining, there may be instances where additional information or knowledge is required to generate accurate and relevant responses. Retrieval-augmented generation (RAG) is a technique that allows LLMs to incorporate knowledge from external sources, such as documents, databases, or APIs, into their generated outputs.

The RAG approach involves three main steps (Figure 4): (1) retrieving relevant data from an external source, (2) augmenting the context of the prompt with the retrieved data, and (3) generating a response based on the prompt and retrieved data using the LLM. As demonstrated by Lewis et al (2020) [7], long documents are split into chunks to fit within the LLM's context window, and vectorized representations of these chunks are stored in a vector database. The query is converted into a vector, and similar chunks are retrieved from the database to augment the prompt. The RAG prompts can be thought of a dynamic prompt which can be used for repeatable tasks (e.g., extract data from the large documents - Table 9), or answer questions (e.g., chatbot assistant).

The RAG prompts also have its limitation, and the chunking, and retrieval process does not consider context, and the search is semantic in nature. The key is to use the relevant information to augment the prompt, and get more accurate response. However, RAG is not free from hallucinations but

prevents it to some extent by using relevant information retrieved from the supplied documents. RAG can often get expensive, as the document size or indexed data size becomes too large, and costly to store.

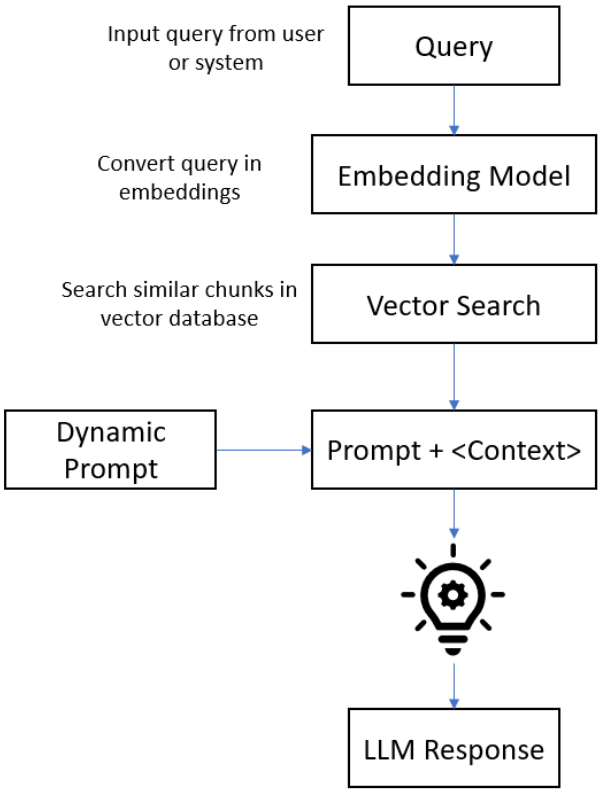

Figure 4. RAG prompt implementation using embeddings

Table 9. Dynamic RAG Prompt – An Example

Prompt (Dynamic)	What are the key features of Tesla Cybertruck, and how does its self-driving control work? [System retrieves relevant information dynamically from a document] Retrieved Context: "The Tesla Cybertruck is an electric pickup truck with a stainless-steel body that was first introduced as a concept in 2019. It's known for its controversial design and is considered to be in a category of its own in the automotive industry. Key features: Design: It has a flat, stainless-steel body. Range: It has an EPA estimated range of 250–340 miles, but an external battery pack can extend the range to over 470 miles. Towing capacity: It has a maximum towing capacity of 11,000 pounds. Payload: It has a payload capacity of up to 2,500 pounds. [+ more text]
LLM Response	[LLM generates a response summarizing the key features and self-driving control based on the prompt and retrieved context]

5.4. Automated Prompt Engineering (APE)

As prompt engineering becomes more complex, there is a growing need to automate the process of crafting and validating prompts. Automated prompt engineering (APE) aims to reduce the human effort involved in prompt engineering by automating various aspects of the process, such as prompt generation, prompt optimization, and prompt evaluation.

APE techniques can involve methods like least-to-most prompting, where the model is encouraged to generate its own prompts by starting with a minimal prompt and iteratively adding more information until the desired output is achieved. Another approach is reasoning without observation (ReWOO), which separates the reasoning process from external observations, allowing the model to optimize its token consumption and potentially improve performance on tasks that require breaking down problems into subproblems.

Table 10. APE Prompt techniques – An example

Prompt	Generate a concise summary of the article on climate change. [LLM generates an initial summary]
Output 1 (Initial summary)	Climate change is causing rising temperatures and sea levels worldwide, impacting ecosystems and weather patterns.
Prompt iteration	Generate a more detailed summary covering key points like causes, impacts, and potential solutions related to climate change. [LLM expands on the summary based on the additional guidance
LLM Response	[More comprehensive summary generated]

6. Conclusions

Prompt engineering has emerged as a critical area of study and application in the field of generative AI, enabling IT professionals and researchers to unlock the full potential of large language models (LLMs) for a wide range of natural language processing tasks. This is a comprehensive overview of various prompt engineering techniques, ranging from basic methods like zero-shot, one-shot, and few-shot prompting, to advanced strategies such as chain-of-thought prompting, self-consistency, tree-of-thoughts prompting, retrieval-augmented generation, and automated prompt engineering.

Through illustrative examples and theoretical insights, we have demonstrated how these techniques can enhance the performance and reliability of LLM deployments across diverse real-world use cases, such as text generation, summarization, question-answering, and decision support systems. By carefully crafting prompts and leveraging the appropriate prompt engineering techniques, IT professionals can effectively guide LLMs towards desired outputs, enabling more robust and efficient generative AI solutions.

While prompt engineering holds immense promise for unlocking the potential of large language models, it is a discipline fraught with challenges that must be navigated skillfully. As this study has demonstrated, achieving consistent and reliable results through prompting requires a delicate balance of specificity and flexibility, as well as a deep understanding of the unique strengths and constraints

of these powerful models. Even seemingly simple tasks can demand extensive experimentation and trial-and-error, highlighting the need for diverse skill sets and multidisciplinary knowledge in this field.

Looking ahead, the demand for sophisticated prompt engineering techniques will only intensify as large language models continue to advance and find application in increasingly complex domains. Addressing issues related to hallucinations, biases, and safety will be paramount, as will the development of more automated, scalable, and adaptive prompting methods. The prompt engineering community must embrace these challenges head-on, fostering a spirit of innovation that can harness the full potential of these models in a responsible and impactful manner.

Future research should prioritize the development of robust and reliable prompting techniques capable of consistently producing accurate and contextually relevant outputs, even for highly specialized tasks. Integrating reinforcement learning algorithms to enable models to adapt their prompting strategies based on experience could be a promising avenue. Additionally, the creation of automated prompt generation frameworks tailored to specific output requirements could streamline the engineering process and broaden accessibility.

Crucially, mitigating hallucinations and biases in model outputs must be a key focus area. Prompt engineering strategies that incorporate external knowledge sources or employ specialized validation prompts could help ensure the trustworthiness and reliability of these models, particularly in high-stakes applications.

As the field of prompt engineering evolves, addressing scalability and adaptability challenges will also be essential. Developing prompting methods that can seamlessly accommodate changes in model architectures, training data, and use cases will be critical to maintaining the relevance and effectiveness of these techniques in the rapidly advancing landscape of generative AI.

By confronting these limitations and charting a course for future research, the prompt engineering community can unlock the full transformative potential of large language models, driving meaningful advancements across a wide range of natural language processing applications. Through sustained innovation and a commitment to responsible development, prompt engineering can shape the future of how we interact with and leverage the power of artificial intelligence.

REFERENCES

[1] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., & Liang, P. (2022). Chain-of-thought prompting elicits reasoning in large language models. arXiv preprint arXiv: 2201.11903.

[2] Kojima, T., Gu, S. S., Reid, M., IE, Y., Shah, R., Liang, P., & Ieong, S. (2022). Large language models are zero-shot reasoners. arXiv preprint arXiv:2205.11916.

[3] Wang, Y., Gan, Z., Wang, J., Zhao, D., Liu, J., & Deng, Z. (2022). Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2212.00398.

[4] Yao, S., Shuster, K., Wong, Y. S., & Weld, D. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv preprint arXiv:2305.10601.

[5] Long, H. (2023). Large Language Model Guided Tree-of-Thought. arXiv preprint arXiv:2305.14952.

[6] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., & Hosseini, H. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

[7] Shin, R., Lin, C. Y., Thomson, S., Chen, O., Kernion, C., Li, P., & Dagan, I. (2022). Automated prompt engineer for large language models. arXiv preprint arXiv:2211.01910.