Prompt Engineering in Generative AI: A Comparative Study and Industry Analysis

Ankur Narwal[1], Alka Kumari[2]

[1]Uttaranchal School of Computing Sciences, Uttaranchal University Dehradun, India
E-mail: ankurnarwal11@gmail.com
[2]Uttaranchal School of Computing Sciences, Uttaranchal University Dehradun -248007, India
E-mail: alkasingh6343@gmail.com

Abstract

In this research, we examine prompt engineering in generative AI from two primary perspectives. First, we examine the performance of several prompting techniques, such as Zero-shot, Few-shot, and Chain-of-Thought. We then explore the practical applications of these strategies in real-world settings. To determine the best tools and timing, we evaluate models such as GPT-4, Deep Seek, and Gemini on tasks including coding, problem-solving, and content summarization. We rely on metrics such as accuracy, token efficiency, and output consistency to measure their effectiveness. We also look at how companies in industries like customer service and education use prompt engineering to improve chat bots and AI tutoring. The results show where each method works well and falls short, pointing to the most useful ways they can be applied.

Keywords: Gen AI, Prompting Techniques, Open AI, chatbot

INTRODUCTION

In the past few years, generative AI has quickly transformed from a specialized area of study to something that many of us use daily, whether it be through virtual assistants, chatbots, or writing tools. This change is the result of a talent known as prompt engineering [1]. Prompt engineering is ultimately just about figuring out how to "talk" to AI in a way that produces the best outcomes.

These models, whether DeepSeek, Gemini, or GPT-4, require some direction, and the way you ask a question can have a significant impact on the response you receive [2].

As these models become smarter and more capable, prompt engineering is becoming even more important. Whether you're building a chatbot, generating code, creating content, or developing an educational tool, the quality of the model's output often comes down to how well it's prompted[3]. Sometimes, even a small tweak in wording or format can completely change the result, which makes prompt design a powerful yet often overlooked part of working with AI[4].

Prompt engineering is still very much a work in progress. There's no single recipe that always does the trick. Methods like Zero-shot, Few-shot, and Chain-of-Thought all have their pros and cons, and the right choice depends on the task and the model you're using [5], [6], [7].

The real challenge is that we don't yet know exactly how these prompts behave once you leave the lab. In

real-world settings, it's not just about getting the right answer. it's about whether the system helps users[8], fits into business goals, and holds up at scale. It's that added complexity that makes prompt engineering such a fascinating and fast-moving field [4], [9].

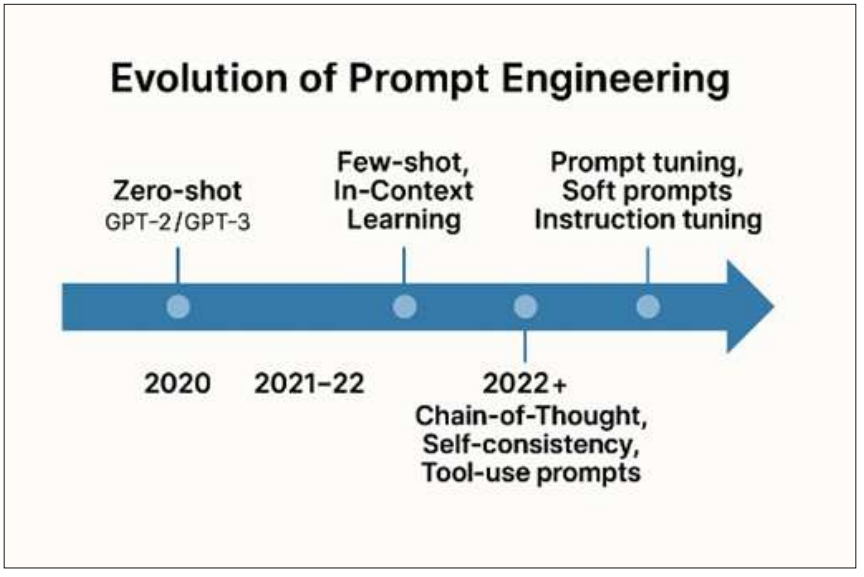

Figure I. *A timeline tracing how prompt engineering has evolved, from simple zero-shot prompting in 2020 to more advanced methods like soft prompting and instruction tuning that began emerging around 2023 and beyond.*

This paper takes a dual-focused approach to explore prompt engineering from both a technical and practical perspective[1]. First, we compare three widely used prompting strategies—Zero-shot, Few-shot, and Chain-of-Thought across tasks like question answering, code generation, and logical reasoning to see how each performs in different contexts[6]. Then, we look at how prompt engineering is being used in real-world applications, from AI-based customer service systems to education platforms[4]. By combining practical experiments with real industry examples, this paper offers a clearer understanding of prompt engineering, both as a technical method and as a growing part of how AI is being applied in everyday tools and services.

LITERATURE REVIEW

A. Understanding Prompting Techniques

Prompt engineering has quickly become something you can't ignore if you're working with large language models (LLMs). As these models become more intelligent, the way you ask a question influences the response you receive. Researchers and developers have therefore devoted a great deal of time to determining the most effective prompting styles. The three that are most frequently discussed are Chain-of-Thought, Few-shot, and Zero-shot.

Zero-shot is the most basic kind of prompting, in which you simply ask a question to the model without providing any examples. For example, "What is Japan's capital?" This is effective for easy tasks, but it may not be sufficient for more complicated ones that require reasoning.[5].

Few-shot prompting gives the model a bit of help. You include a few examples first, so it can pick up on the pattern before answering. It's more reliable than zero-shot in many cases, but the results still depend a lot on which examples you give and how you present them[10].

Chain-of-Thought (CoT) prompting involves having the model go through its reasoning step by step. This makes it particularly helpful when

tackling logical or mathematical problems, when the method is just as important as the solution. A 2022 study led by Jason Wei showed that this approach led to much better results on math benchmarks like GSM8K, especially when used with larger models like GPT-4 and PaLM.[6].

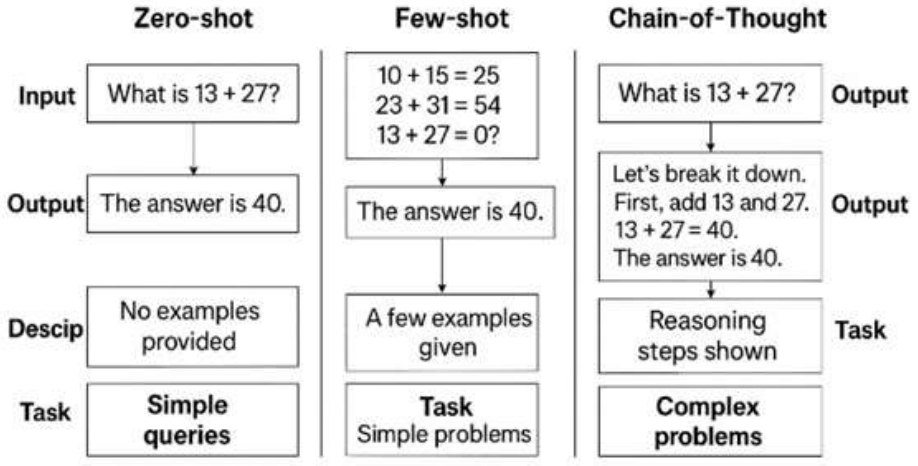

Figure II. *Comparison of Zero-shot, Few-shot, and Chain-of-Thought prompting techniques with example structures and tasks.*

Every method brings something unique. The best choice depends on the task and the model you use, as one approach may not be suitable for all [3], [7].

B. Benchmarks and Experimental Evidence

To understand how different prompting strategies work in real-life situations, researchers have put them to the test. OpenAI's GPT-3 paper[10], *"Language Models are Few-Shot Learners," found that giving the model just a few examples, known as few-shot prompting, helped it perform surprisingly well on tasks such as translation and question-answering, even without any additional* training. Later, a 2022 study called *Chain of Thought Prompting Elicits Reasoning in Large Language Models* by Jason Wei and his team took this a step further[6]. They showed that when the model is asked to explain its thinking step by step—using a method called Chain-of-Thought prompting—it does much better on more complex problems, especially when using larger models like PaLM and GPT-4.

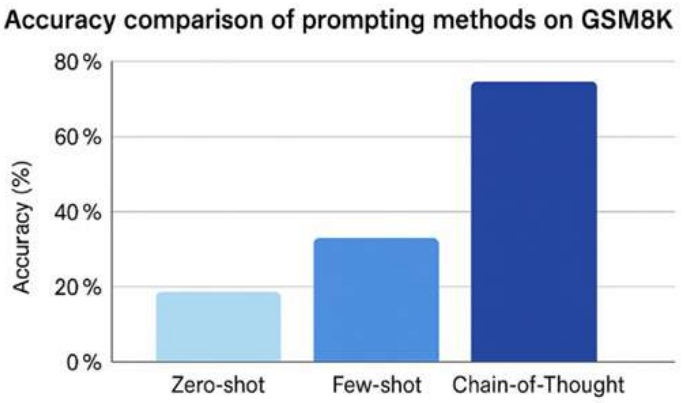

Figure III. *Accuracy comparison of prompting methods on the GSM8K benchmark from Wei et al. (2022).*

Some research has also focused on real-world challenges. For instance, in *Large Language Models Are Zero-Shot Reasoners*, Tomoya Kojima and his team found that Chain-of-Thought prompting tends to make responses longer[5], [11]. That might not sound like a big deal, but in real-world applications, it can slow things down or even drive up costs. Other studies comparing GPT-4 with open-source models like DeepSeek and LLaMA 3 show that the same prompting technique doesn't always work the same way[11]. Each model responds a bit differently, depending on how it was trained, so there's no one-size-fits-all approach[12].

C. Prompt Engineering in Real-World Applications

Outside research labs, prompt engineering has a big role in helping industries use AI. In customer service, businesses design specific prompts to make chatbots respond with empathy and accuracy [13]. For example, Intercom relies on prompt chains to help AI grasp what users mean and follow company rules.

Creative tools like Jasper or Notion AI also use prompt layering, multiple prompts stacked or refined in sequence, to maintain tone, style, and context [14]. These real-world implementations often depend on trial-and-error refinement, rather than academic benchmarks, highlighting the difference between theoretical performance and production effectiveness [15].

D. Gaps between Technical and Practical Use

Despite all the available research, one thing stands out. There's a difference between how prompting is supposed to work in theory and how people use it. Researchers usually emphasize benchmarks and precise results[16]. Businesses, however, often prioritize tone, dependability, customer happiness, and keeping costs low. Real-world prompting tends to be messier, more iterative, more context-driven, and harder to evaluate with standard metrics.

Moreover, most existing studies evaluate prompts on isolated tasks, rather than in full workflows. There's little research that connects prompt design to long-term product performance or user feedback[17]. That makes it harder for developers or businesses to know which prompting strategy to choose or how to scale it effectively.

This paper aims to bridge that gap. By combining experimental comparisons of prompting techniques with an analysis of how they're used in real systems, we hope to bring together both sides of the conversation technical performance and practical impact[12].

METHODOLOGY

A. Comparative Analysis of Prompting Techniques

This part of the study explores how different prompting strategies perform by comparing three widely used methods: **Zero-shot**, **Few-shot**, and **Chain-of-Thought (CoT)**. These techniques are tested across several well-known generative AI models and a variety of tasks to see how they hold up in different scenarios.

Prompt Styles

- **Zero-shot prompting** tells the model to follow directions without providing examples for guidance. The job depends on the model's existing knowledge to work things out[5].
- **Few-shot prompting** provides a few examples of how inputs and outputs should look. These examples help the model notice patterns and answer in a similar way[10].
- **Chain-of-Thought (CoT)**By asking the model to explain its reasoning in detail before responding, prompting provides additional guidance. This is effective for tasks requiring logical reasoning.[6].

Models Compared

The performance of the following large language models is examined in the analysis:

- GPT-4 (OpenAI)
- DeepSeek V2
- Gemini 1.5 Pro (Google)

Types of Tasks

We test each combination of the model and prompt method in four key areas:

- **Math Word Problems** — we use datasets like GSM8K that need step-by-step thinking and logical problem-solving[18].

- **Coding Challenges** — Tasks include basic coding in Python or JavaScript, like finding factorials, creating Fibonacci sequences, or sorting lists[19].
- **Summarization** — condensing lengthy texts into concise versions while retaining the main ideas[20].
- **Question Answering** — Giving straight-to-the-point or factual replies based on the provided text.

How We're Measuring Performance

We evaluate each approach using a fair and consistent process.

- **Accuracy** - Is the information provided clear and correct?
- **Token Efficiency** - How helpful is the response, and how well does it balance the word count?
- **Robustness** - can the system handle small changes in the wording of the question while still working well?[21]

B. Real-World Case Study: Prompt Engineering's use in industry.

This section examines the ways prompt engineering is used in teaching, customer service, and creative tools outside of research settings. It extends beyond controlled experiments to demonstrate practical use when users interact with these devices daily [8].

Consider schools. Khanmigo, created by Khan Academy, helps students learn logical reasoning and walks them through every step of the process while also solving math problems[14]. Customer support tools like Intercom use smart prompts to figure out user questions and give answers that match company rules[22]. Creative programs like Jasper and Notion AI focus on matching the tone and style of content so it feels like it came from the company[23].Writing good prompts is not just about technical accuracy. Good prompts require clarity towards your idea and also the skills of using the AI tool.

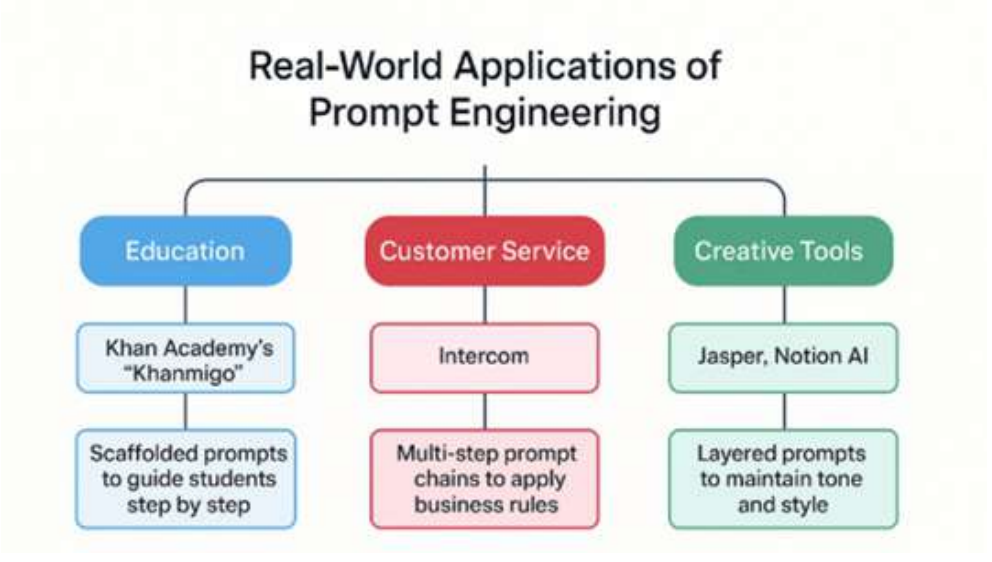

Figure IV. *Prompting strategies used in real-world tools across education, customer service, and content creation are based on how actual products guide conversations and generate responses.*

RESULTS

When we tested Zero-shot, Few-shot, and Chain-of-Thought prompting, some clear patterns stood out. Each method had its strengths depending on the task and the model, and here are some of the most important takeaways:

Task Type	Prompting Method	GPT-4	DeepSeek	Gemini
Math Problems	Zero-shot	52%	39%	42%
	Few-shot	67%	54%	59%
	Chain-of-Thought	**85%**	71%	77%
Coding Challenges	Zero-shot	61%	50%	55%
	Few-shot	73%	63%	68%
	Chain-of-Thought	**79%**	67%	72%
Summarization	Zero-shot	70%	62%	66%
	Few-shot	**78%**	69%	72%
	Chain-of-Thought	75%	68%	71%
Question Answering	Zero-shot	82%	70%	74%
	Few-shot	**87%**	77%	81%
	Chain-of-Thought	84%	75%	79%

Figure V. *A side-by-side look at how different prompt styles perform across common tasks like solving problems, writing code, summarizing text, and answering questions.*

DISCUSSION

A. When is a few-shot better than Zero-shot?

Few-shot prompting showed better results than Zero-shot in every task studied here[10]. The main reason is that Few-shot examples allow models to build context and understand the structure of responses more[5]. This becomes even more noticeable with tasks such as summarization and question answering, where keeping a consistent format and tone plays an important role. Even for simpler math or code problems, seeing a few structured examples helps guide the model's response more reliably than abstract instructions. However, Few-shot does come at the cost of **token length** and sometimes **reduced flexibility**, making it less suitable when low latency or token limits are priorities [24].

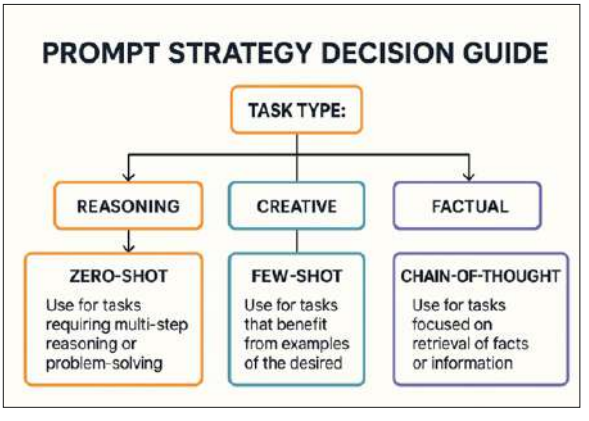

Figure VI. *A simple guide to choosing the right prompting approach based on the type of task whether it's creative, factual, or reasoning-based*

B. Is Chain-of-Thought Always Helpful?

Chain-of-Thought prompting fits well when a task needs careful reasoning, like doing math or writing code[6]. It helps break problems into smaller steps, which makes the thought process clearer. This approach can also lower the chances of missing important parts. When solving math problems using CoT, accuracy improves by more than 20% compared to Zero-shot methods[5].

However, CoT does not work better in every case. Few-shot prompting can sometimes do a better job at tasks like giving summaries or answering questions. CoT's "thinking out loud" process may add extra details, which might not be helpful for short and simple replies where efficiency matters more[25].

C. Gaps between Theory and Practice

There's still a gap between results from labs and real-world usage. Researchers focus on how precise their methods are[14], but businesses care about different things. They care if the answers seem natural, meet a goal, reflect their brand's style, and are cheap to implement [26].

Those working in the field learn by experimenting with prompts and tweaking them as they go. They keep improving them based on what users say. Benchmarks cannot show this type of learning as it occurs in real time. To address this issue, better ways are needed to judge the quality of prompts. These ways should look at not just accuracy but also how clear the results are, how useful they are, and whether they match the intended goal [25].

CONCLUSION

This paper examined prompt engineering by exploring its technical aspects and its use in real-life applications. It compared various methods of prompting and how industries apply them. Tests

on models like GPT-4, DeepSeek, and Gemini showed that Chain-of-Thought prompting performs well. COT is good at handling tasks that involve reasoning.

Few-shot prompts work better when tasks require clear answers or a specific structure. It's often used to summarize text or answer direct questions. This approach shows up both in practical tools and research projects. For instance, Khan Academy's Khanmigo and Intercom use Chain-of-Thought prompts to guide users step-by-step. This helps users think and understand the reasoning behind each part.

Meanwhile, platforms like Jasper use Few-shot prompting to keep the tone and style of their content steady.

Prompt engineering in real-life situations tends to be more complicated than what academic tests show. It requires ongoing tweaking of prompts, experimenting, adding step-by-step instructions, and preparing backup strategies to deal with surprising replies.

References

1. X. Amatriain, "Prompt Design and Engineering: Introduction and Advanced Methods," May 2024, [Online]. Available: http://arxiv.org/abs/2401.14423
2. S. Bubeck *et al.*, "Sparks of Artificial General Intelligence: Early experiments with GPT-4," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2303.12712
3. L. Reynolds and K. McDonell, "Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.07350
4. M. Desmond and M. Brachman, "Exploring Prompt Engineering Practices in the Enterprise," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2403.08950
5. T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large Language Models are Zero-Shot Reasoners," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2205.11916
6. J. Wei *et al.*, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 2023, [Online]. Available: http://arxiv.org/abs/2201.11903
7. P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing," *ACM Comput Surv*, vol. 55, no. 9, Sep. 2023, doi: 10.1145/3560815.
8. P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications," Mar. 2025, [Online]. Available: http://arxiv.org/abs/2402.07927
9. K. Singhal *et al.*, "Large language models encode clinical knowledge," *Nature*, vol. 620, no. 7972, pp. 172–180, Aug. 2023, doi: 10.1038/s41586-023-06291-2.
10. T. B. Brown *et al.*, "Language Models are Few-Shot Learners," Jul. 2020, [Online]. Available: http://arxiv.org/abs/2005.14165
11. H. Touvron *et al.*, "Llama 2: Open Foundation and Fine-Tuned Chat Models," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.09288
12. P. Liang *et al.*, "Holistic Evaluation of Language Models," Oct. 2023, [Online]. Available: http://arxiv.org/abs/2211.09110
13. Y. Zhou *et al.*, "Large Language Models Are Human-Level Prompt Engineers," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2211.01910
14. Q. Lyu *et al.*, "Faithful Chain-of-Thought Reasoning," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2301.13379
15. J. Kang, W. Xu, and A. Ritter, "Distill or Annotate? Cost-Efficient Fine-Tuning of Compact Models," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2305.01645
16. C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," Sep. 2023, [Online]. Available: http://arxiv.org/abs/1910.10683
17. Y. Bai *et al.*, "Constitutional AI: Harmlessness from AI Feedback," Dec. 2022, [Online]. Available: http://arxiv.org/abs/2212.08073
18. K. Cobbe *et al.*, "Training Verifiers to Solve Math Word Problems," Nov. 2021, [Online]. Available: http://arxiv.org/abs/2110.14168
19. M. Chen *et al.*, "Evaluating Large Language Models Trained on Code," Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03374
20. J. W. Rae *et al.*, "Scaling Language Models: Methods, Analysis & Insights from Training Gopher," Jan. 2022, [Online]. Available: http://arxiv.org/abs/2112.11446

21. OpenAI *et al.*, "GPT-4 Technical Report," Mar. 2024, [Online]. Available: http://arxiv.org/abs/2303.08774
22. C. Qin, A. Zhang, Z. Zhang, J. Chen, M. Yasunaga, and D. Yang, "Is ChatGPT a General-Purpose Natural Language Processing Task Solver?" Nov. 2023, [Online]. Available: http://arxiv.org/abs/2302.06476
23. A. Bolotin, "The paradox of classical reasoning," Jul. 2022, doi: 10.1007/s10701-022-00604-7.
24. V. Sanh *et al.*, "Multitask Prompted Training Enables Zero-Shot Task Generalization," Mar. 2022, [Online]. Available: http://arxiv.org/abs/2110.08207
25. D. Zhou *et al.*, "Least-to-Most Prompting Enables Complex Reasoning in Large Language Models," Apr. 2023, [Online]. Available: http://arxiv.org/abs/2205.10625
26. A. Tamkin, M. Brundage, J. Clark, and D. Ganguli, "Understanding the Capabilities, Limitations, and Societal Impact of Large Language Models," Feb. 2021, [Online]. Available: http://arxiv.org/abs/2102.02503