International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

The Power of Prompt Engineering: Refining Human - AI Interaction with Large Language Models in The Field of Engineering

Satish Kathiriya[1], Mahidhar Mullapudi [2], Abhishek Shende[3]

[1]Software Engineer, CA, USA

[2]Senior Software Engineer, WA, USA

[3]Senior Principal Software Engineer, CA, USA

Abstract: ***This paper presents a pioneering exploration into prompt engineering, a critical aspect of interactions with Large Language Models (LLMs) in engineering contexts. We delve into the nuanced strategies and methodologies of prompt engineering, demonstrating its profound impact on the performance and reliability of LLMs like GPT - 4. We explored a range of techniques, from simple, clear instructions to more complex few - shot learning prompts and evaluated their effectiveness in various scenarios. The findings highlight the significant potential of prompt engineering in enhancing the precision, efficiency, and applicability of LLMs in engineering, setting a foundation for future advancements in human - AI collaboration.***

Keywords: Large Language Model, Prompt Pattern, Prompt Engineering, Human - AI Interaction, Few - shot Learning, GPT - 4, Foundation Model

1. Introduction

The advent of Large Language Models (LLMs) like GPT - 4 has revolutionized the field of artificial intelligence, offering unprecedented capabilities in natural language understanding and generation. However, the efficacy of these models in practical applications hinges significantly on the art of 'prompt engineering', the method of crafting inputs to elicit the most effective outputs from LLMs [1]. This paper explores the intricacies and significance of prompt engineering, particularly in engineering contexts, where precision and reliability are paramount.

We begin by examining the evolution of LLMs, with a focus on their application in engineering disciplines. The core of our investigation delves into the various methodologies of prompt engineering, ranging from simple directives to sophisticated few - shot learning prompts. We analyze how different prompting techniques can drastically alter the performance of LLMs, using GPT - 4 as a primary case study.

Furthermore, the paper investigates novel prompt engineering strategies tailored for specific engineering applications, offering insights into how these methods can optimize the utility of LLMs in practical scenarios. By conducting a series of experiments and evaluations, we present empirical evidence of the effectiveness of these techniques.

In conclusion, this research not only underscores the critical role of prompt engineering in maximizing the potential of LLMs but also paves the way for future innovations in human - AI collaboration within the engineering domain.

2. Foundation Models

Foundation models are large machine learning (ML) models pre - trained on vast datasets. They are versatile, capable of performing a wide range of general tasks such as language understanding, text and image generation, and conversational interactions [6]. Over the years, these models have evolved significantly in terms of size and complexity. For instance, BERT, a bidirectional foundation model released in 2018, used 340 million parameters [7], whereas GPT - 4, released by OpenAI in 2023 [6], operates on a staggering 170 trillion parameters.

The importance of foundation models lies in their ability to transform the machine learning lifecycle. They are expensive to develop but offer a long - term, cost - effective solution for data scientists. Instead of building unique ML models from scratch, pre - trained foundation models can be adapted for new applications, significantly reducing development time and costs [6].

- **Large - scale Pre - training:** Foundation models are colossal in size and are pre - trained on extensive datasets. This training encompasses a wide range of topics, equipping the models with a broad understanding of numerous subjects.
- **Adaptability to Specialized Tasks:** Despite their generalized training, these models possess the unique ability to be fine - tuned for specific, more specialized tasks, making them highly versatile.
- **Expanding Beyond Text:** Initially, foundation models were predominantly trained on text data. However, recent advancements have enabled them to evolve into multimodal domains, meaning they can now process and understand a combination of different data types like text, images, and sounds.
- **Learning Contextual Probabilities:** A significant aspect of these models is their ability to learn the likelihood of certain words appearing in specific contexts. This capability is crucial for understanding and generating human - like text.

- **Training Objective:** The models are often trained with tasks like predicting a missing word in a given text sequence. This method, also known as masked language modeling, helps the model learn contextual relationships and nuances in language.

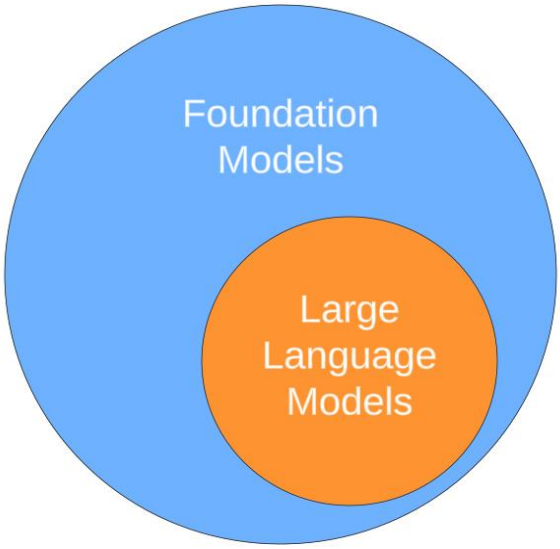

Figure 1: Foundation Model and Large Language Model

2.2 How Foundation Models Work

Foundation models, such as large language models (LLMs), operate on principles of generative artificial intelligence. They use neural networks, including generative adversarial networks (GANs), transformers, and variational encoders, to predict the next item in a sequence based on learned patterns and relationships. This self - supervised learning approach is what differentiates LLMs from previous ML architectures, which relied on supervised or unsupervised learning [7].

2.3 Capabilities of Foundation Models

Foundation models have a range of capabilities, including:

- **Language Processing**: Answering questions, writing scripts or articles, and language translation.
- **Visual Comprehension**: Image identification, autonomous driving, and robotics applications.
- **Code Generation**: Generating and evaluating code in various programming languages.
- **Human - Centered Engagement**: Supporting human decision - making in fields like healthcare and analytics.

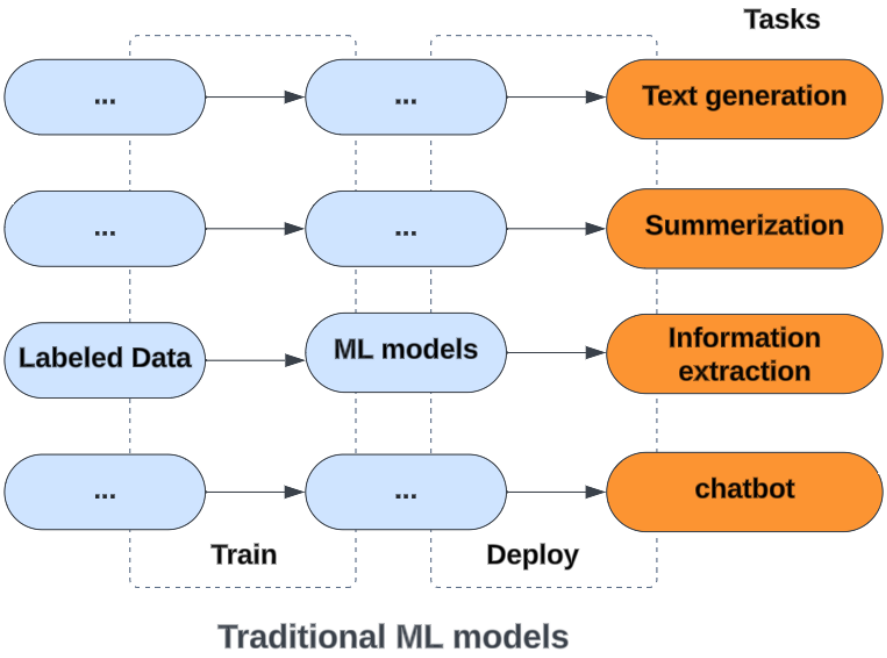

Figure 2: Traditional ML Model Use Cases

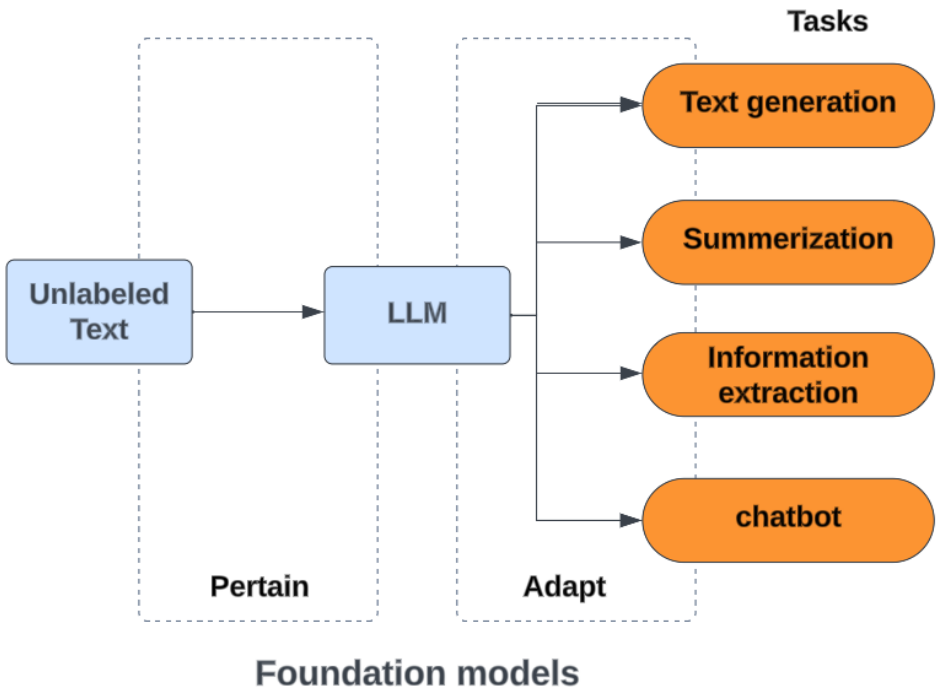

Figure 3: Foundation Model Use Cases

2.4 Traditional Models vs. Foundation Models

Table 1: Comparison of Foundation Model and Traditional Model

Feature/Aspect	Foundation Models	Traditional Models
Data Scale and Diversity	Trained on vast, diverse datasets (including multimodal)	Typically trained on smaller, specific datasets
Flexibility and Adaptability	Highly adaptable to various specialized tasks	Limited flexibility, often task - specific
Training Objective	Learning context and predicting elements (like words)	Often focused on specific tasks (e. g., classification)
Learning Approach	Learn probabilities of elements in context (e. g., words in text)	More rigid learning rules, less context - focused
Application Scope	Broad and versatile across numerous domains	Generally narrower, specific to the trained task
Resource Intensity	Resource - intensive due to size and complexity	Less resource - intensive, smaller in scale

Foundation models like GPT - 4, Claude, Amazon Titan and Bloom represent the apex of this evolutionary journey in ML. Their unparalleled parameter scale and sophisticated training mechanisms highlight the incredible progress made in the field. This evolution underscores the critical role of prompt engineering in harnessing the full potential of these models. Prompt engineering, the art of designing inputs to elicit desired outputs from these models, has become increasingly important. It allows for the effective application of these advanced models in various domains, demonstrating their transformative impact in technology and society at large.

3. GPT - 4:

In the wake of our introduction's context, we focus on the technological marvel, GPT - 4, developed by OpenAI [8]. This model is trained on a vast corpus of text data, boasting a parameter count exponentially larger than its predecessor, GPT - 3, which had 175 billion parameters [16]. This immense scale significantly enhances its language processing capabilities. All of the output in the following sections are generated by GPT - 4, developed by OpenAI [8].

GPT - 4's architectural bedrock is the transformer model [17], a sophisticated mechanism that revolutionizes how inputs are processed. Unlike traditional neural networks, transformers employ attention mechanisms to assign context - sensitive weights to input data, enabling a more nuanced understanding and generation of language.

The operational process begins with the model converting input prompts into tokens, the fundamental units of language it can interpret. These tokens pass through transformer layers, where their interrelations and context are intricately analyzed. The attention mechanisms within these layers adeptly weigh the tokens, forming comprehensive intermediate representations of the input data. This process culminates in the generation of coherent, contextually relevant text.

A pivotal aspect of GPT - 4's functionality is its application of Reinforcement Learning from Human Feedback (RLHF) [22, 23]. This technique leverages human feedback as a reward mechanism, refining the model's capability to adhere to a diverse range of written instructions with heightened accuracy.

The model's output variability is managed through a randomness function, influenced by two primary parameters: temperature and top - k sampling [23]. The temperature setting controls the balance between randomness and determinism in the responses. A higher temperature induces variability, while a lower one leans towards predictability.

Top - k sampling, on the other hand, restricts the model's choices to the top 'k' probable tokens during output generation, adding another layer of refinement.

As we proceed, the prowess of GPT - 4 will be demonstrated through various applications and scenarios, underscoring its significance in our exploration of prompt engineering in the field of engineering.

3.1. Prompt Engineering

In the context of advancements in Large Language Models like GPT - 4, prompt engineering emerges as a transformative practice, shaping how these AI systems interpret and respond to textual queries. This is particularly true for GPT - 4, whose advanced features, such as a vast training dataset and sophisticated transformer architecture, make it especially receptive to nuanced prompt design [7]. Whether in complex engineering problem - solving, code generation, or content summarization, the artful construction of textual cues is fundamental in eliciting high - quality, contextually relevant responses from the model [10].

Unlike traditional machine learning models, which often require extensive retraining for different tasks, LLMs like GPT - 4 provide a dynamic and flexible approach through prompt engineering. This adaptability is invaluable in engineering, enabling rapid customization of model outputs to specific challenges, leading to groundbreaking insights [12]. A well - constructed prompt acts as a conduit, focusing the model's extensive knowledge and computational power into task - specific responses. The systematic design and optimization of these prompts are crucial in guiding LLM responses, ensuring accuracy, relevance, and coherence.

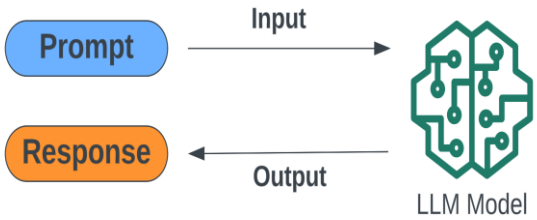

Figure 4: Prompt and response in LLM Model

The process of crafting effective prompts is iterative, often necessitating multiple refinements to achieve the optimal balance. The quality and structure of these prompts significantly influence LLM performance, with skillfully constructed prompts crucial for eliciting precise responses and counteracting potential inaccuracies or hallucinations in the model's output [10] [11]. Prompt engineering encompasses a spectrum of approaches, from established best practices to innovative research techniques, each acting as a tailored input to define tasks, set constraints, provide examples, or specify the desired response format, thereby shaping the final outcome [11] [12].

This section delves into the core elements of effective prompt engineering. We explore strategic prompt structuring, leveraging GPT - 4's inherent capabilities to yield responses that are both accurate and contextually nuanced. Techniques such as role - prompting, where the model assumes a specific 'role' for the task, and iterative refinement, where prompts are progressively adjusted based on initial responses, are examined in detail [13, 14]. These methods equip researchers and practitioners with the knowledge to effectively utilize LLMs, pushing the boundaries of their application in engineering and beyond.

Through practical examples and hypothetical scenarios, we will illustrate how subtle variations in prompt construction can significantly influence the model's output, demonstrating the transformative impact of prompt engineering in harnessing the full potential of LLMs for diverse engineering challenges.

3.2 Elements of a prompt

Prompt construction is an intricate process involving several key components: Instruction, Context, Input Data, and Output Indicator [20]. These elements are essential in guiding LLMs to generate precise and relevant responses [13, 19]. To illustrate their significance, we consider an example within the domain of coding and software engineering.

1) Instruction: This is the directive or command that tells the LLM what task to perform. In our case, the instruction could be "Write a Python function"
2) Context: This provides additional information or background needed to understand the task. For coding, the context might include specifying the programming language or explaining any specific requirements of the function.
3) Input Data: This is the specific information or parameters that the LLM needs to perform the task. In our Python coding example, the input data could be a formula with which it feels like temperature is to be calculated.
4) Output Indicator: This tells the LLM how to format or present its response. For a coding task, the output indicator might specify that the response should be in the form of executable Python code, which should be the input parameter for the function with comments explaining each step.

Following example in figure 5 demonstrates how each component plays a pivotal role in the construction of an effective prompt. By carefully crafting each element, users can guide LLMs like GPT - 4 to produce outputs that are not only technically accurate but also contextually relevant and tailored to specific requirements. As we explore further in this paper, the nuanced application of these elements is central to successful prompt engineering across various disciplines.

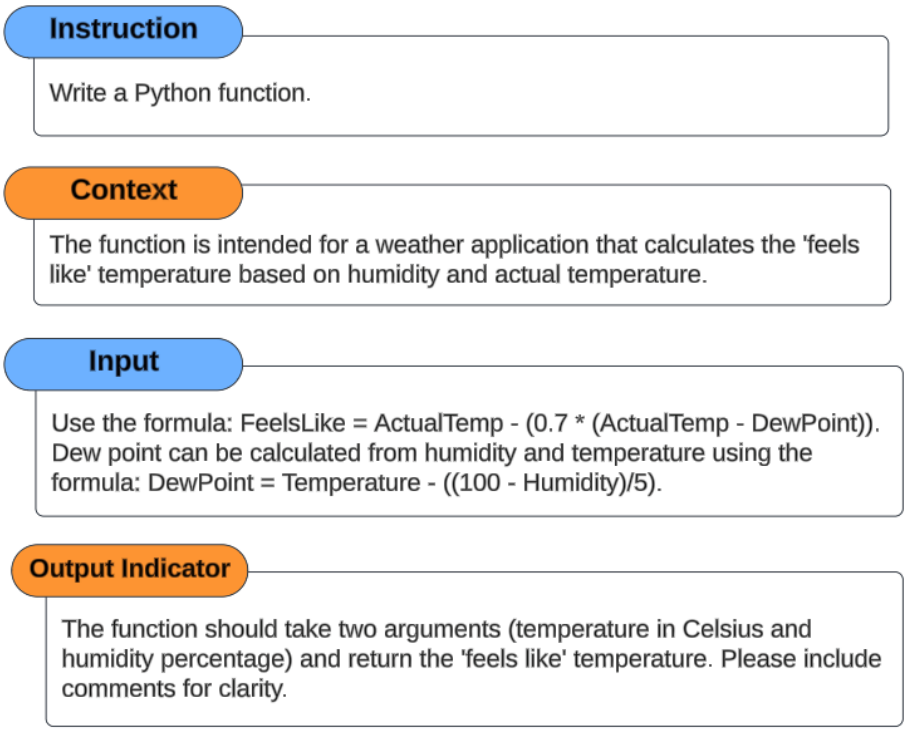

Figure 5: Prompt Engineering Example to Write Python Code

Crafting the Prompt:
"Can you write a Python function for a weather application that calculates the 'feels like' temperature? The function should use the formulas: FeelsLike = ActualTemp - (0.7 * (ActualTemp - DewPoint)) and DewPoint = Temperature - ((100 - Humidity) /5). It needs to take two arguments: temperature in Celsius and humidity percentage, and then return the 'feels like' temperature. Make sure to include comments in the function for clarity. "

Generated output:

```python
def calculate_feels_like_temperature(temperature, humidity):
    """
    Calculate the 'feels like' temperature based on actual temperature and humidity.

    Args:
    temperature (float): Actual temperature in Celsius.
    humidity (float): Humidity percentage.

    Returns:
    float: The calculated 'feels like' temperature.
    """

    # Calculate the dew point
    dew_point = temperature - ((100 - humidity) / 5)

    # Calculate the 'feels like' temperature
    feels_like_temp = temperature - (0.7 * (temperature - dew_point))

    return feels_like_temp

# Example usage
calculate_feels_like_temperature(30, 50)
```

Figure 6: Python Code Generated using Prompt Engineering

4. Techniques

In this section, we build on the foundational aspects of prompt construction discussed earlier to explore specific techniques that enhance the efficacy of Large Language Models (LLMs) in practical scenarios. These techniques are integral to fine - tuning the model's response and ensuring relevance and precision in its outputs.

4.1 Giving clear and precise instructions

The technique of giving clear and precise instructions in prompt engineering mirrors effective communication

strategies commonly used in teaching or instructing. It involves crafting prompts that provide clear, contextualized information, enabling the Large Language Model (LLM) to generate focused and relevant responses. This method narrows down the model's scope of response by specifying the context and the specific information required. It has been observed that the output generated by GPT - 4, introduced in Section 2, tends to be excessively general when provided with fundamental instructions devoid of any supplementary description [19]. An example prompt is shown in Figure 4. When the model is prompted with basic instruction, it faces a plethora of options, which makes the result quite broad. Hence, a comprehensive description is imperative to elicit more precise and relevant outputs [28].

This response illustrates an issue with a prompt that lacks specificity and clarity in its request. The prompt asks for current trends in software engineering, a broad and extensive topic, which results in a response that, while accurate, is overly general and possibly overwhelming with the breadth of topics covered. The response jumps across various significant trends without offering a focused or in - depth exploration of any single trend.

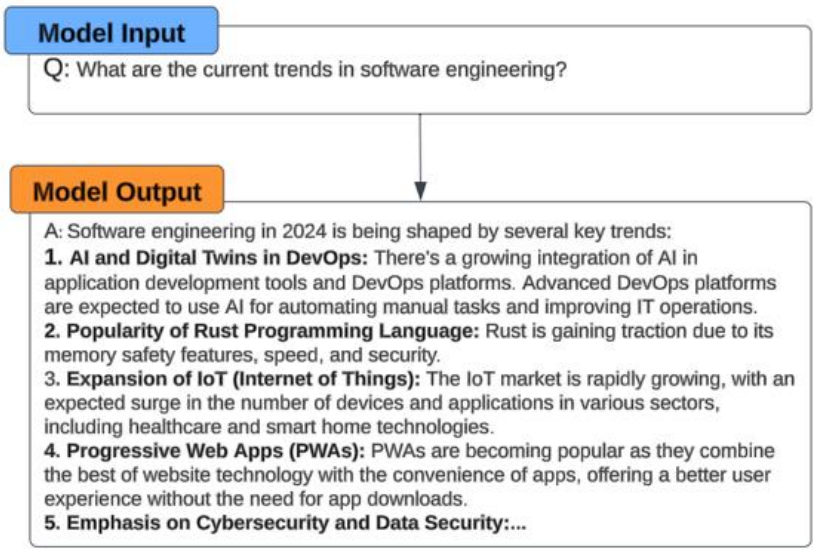

Figure 7: Instruction with Unclear and Vague Prompt

This lack of precision in the prompt leads to a response that, although informative, may not effectively meet the specific needs or interests of the reader. For instance, if the intent was to understand the impact of these trends on a particular area of software engineering, such as artificial intelligence in software engineering, particularly in the areas of automated code review and bug detection, the response does not provide this targeted insight. Additionally, the broad nature of the response fails to delve into the intricacies of each trend, such as the challenges and opportunities they present to software engineers.

A more effective prompt would narrow the focus to a specific aspect of software engineering or request detailed insights into how a particular trend is influencing the field. This would guide the response to be more targeted and relevant, providing a clearer and more concise understanding of the specified area within software engineering.

Prompt in Figure 8 specifically focuses on the application of AI in software engineering, narrowing down to automated code review and bug detection. It asks for a comparison with traditional methods, providing a clear direction for the analysis.

Through these examples, we demonstrate the importance of crafting detailed and contextualized prompts. This approach is crucial in directing the LLM's response towards the desired area of focus, ensuring that the outputs are not only accurate but also relevant to the specific query.

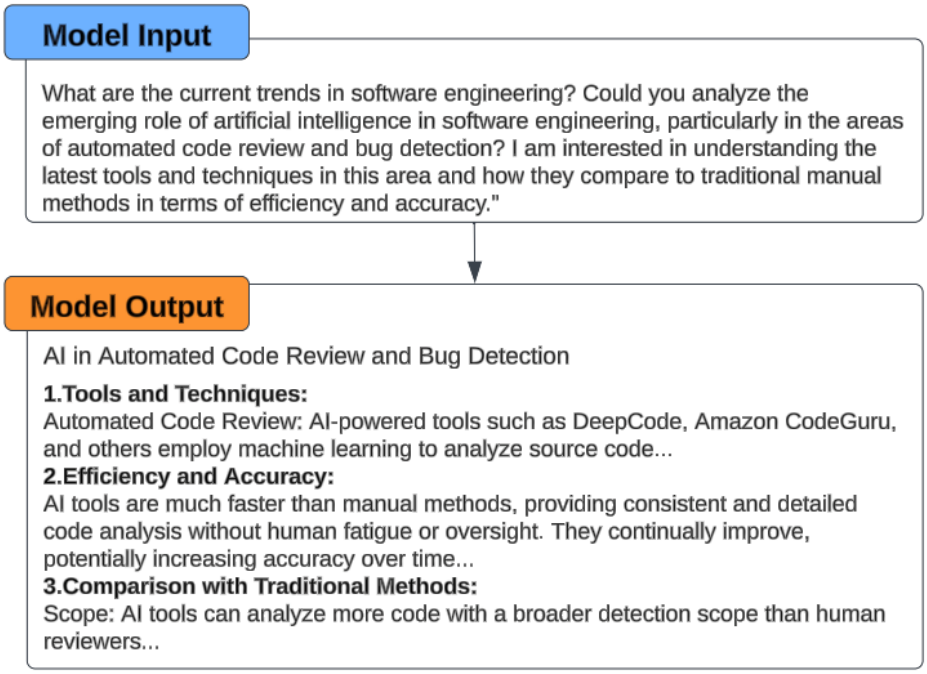

Figure 8: Instruction with Clear and Precise Prompt

4.2 Role - prompting

Role - prompting is a strategic method in prompt engineering where the Large Language Model (LLM) is assigned a specific role or persona to guide its responses. This technique is effective in aligning the model's output with particular thematic or professional requirements [15]. By instructing the model to assume a specific role, such as an expert in a certain field or a character from a specific time period, the responses can be tailored to reflect the knowledge, tone, and style appropriate for that role [21].

4.2.1 Enhancing Contextual Relevance:

When the model adopts a specific role, it draws upon relevant knowledge and mimics the communication style associated with that role. This leads to responses that are not only contextually accurate but also stylistically appropriate.

4.2.2 Versatility in Applications:

Role - prompting can be particularly useful in diverse fields, including education, professional advice, creative writing, and technical support. It allows the model to provide specialized responses that go beyond general information.

4.2.3 Example

In an engineering context, role - prompting can be used to simulate expert advice or to provide specialized insights. For instance, a prompt given in Figure 9, prompting the LLM to act as a civil engineer can yield detailed and technically sound advice on construction and infrastructure.

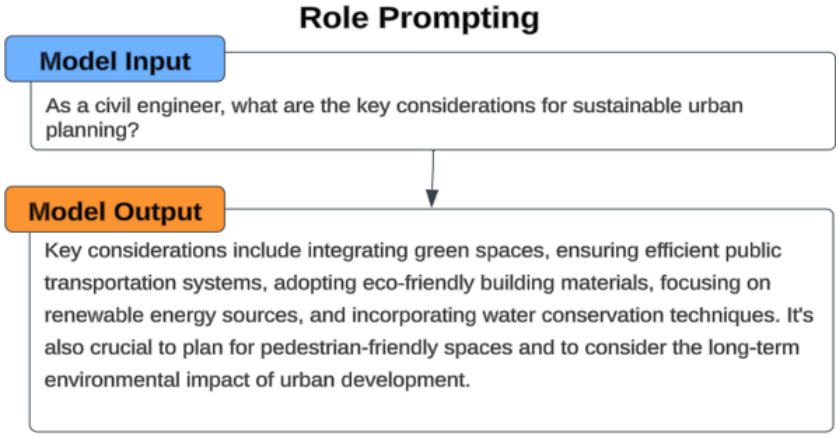

Figure 9: Role Prompt Example

Role - prompting can also be creatively adapted to non - professional scenarios, such as asking the model to assume the role of a historical figure for educational purposes or a fictional character for creative writing. This versatility makes role - prompting a powerful tool in prompt engineering.

4.2.4 Considerations

When using role - prompting, it's important to clearly define the role and context to ensure that the model's responses are aligned with the intended perspective. The effectiveness of this technique largely depends on the model's training and its knowledge base related to the chosen role.

4.3 Zero - shot prompting

Zero - shot prompting is a technique where the model is presented with a task or question without any accompanying examples. This method relies on the model's pre - existing knowledge and its ability to infer the correct approach based solely on the prompt. Zero - shot prompting is particularly useful when the task is within the model's strong knowledge domains or when the nature of the task is such that it does not benefit from additional context provided by examples.

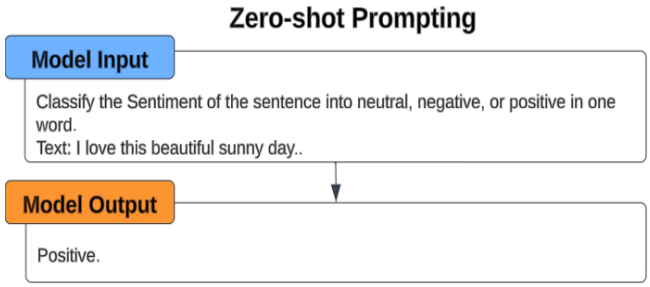

Figure 10: Zero Shot Prompt Example

Prompt given in Figure 10 is the task of sentiment analysis: This prompt exemplifies zero - shot prompting, as it does not offer examples but relies on the model's capacity to comprehend and classify sentiment based on its pre - trained knowledge. The model's response, typically a concise sentiment classification, underscores its proficiency in applying learned patterns to specific tasks without additional guidance.

Despite its efficiency, zero - shot prompting's success hinges on the complexity of the task and the model's training depth. For specialized or intricate topics, the responses may lack precision, necessitating the use of one - shot or few - shot prompting techniques. However, in general inquiries or areas with well - established knowledge, zero - shot prompting remains a swift and effective means to engage the model's capabilities.

4.4 One - shot prompting

One - shot prompting is an intermediate approach between zero - shot and few - shot prompting. In this technique, the model is given a single example with instructions to guide its understanding of a task [21]. This approach is particularly useful for tasks that the model has not been explicitly trained on [22]. The single example acts as a blueprint, enabling the model to extrapolate from it and generate relevant outputs in a similar format or context.

Figure 11: One Shot Prompt

Example: Consider a task where a language model, not specifically trained in culinary content, is asked to generate recipes. Even though the model hasn't seen this specific example during training, it can use the structure of the provided example to generate a new recipe:

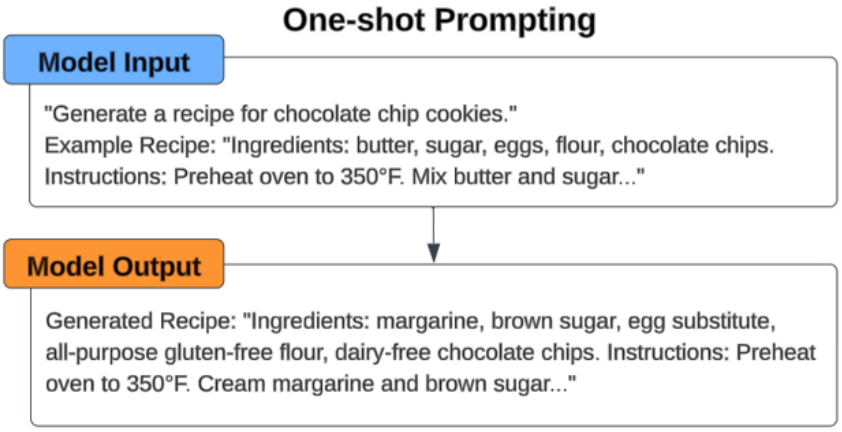

Figure 12: One Shot Prompt Example

In this case, the model uses the one - shot example to understand the desired output format and generate a new recipe based on that understanding.

One - shot prompting is effective for tasks where a single, well - chosen example can provide sufficient structure and context. It's crucial to select an example that is representative of the desired output. The quality and relevance of this example greatly influence the model's ability to generalize and produce accurate responses. However, this approach can be limited if the single example does not encapsulate the full range of desired outputs or if it's too specific, leading the model to generate overly narrow responses. Despite these limitations, one - shot prompting offers a balance between no prior examples (zero - shot) and the need for multiple examples (few - shot), making it a versatile tool in prompt engineering.

4.5 Few - shot prompting

Few - shot prompting is a method where a model is provided with a small number of examples along with instructions to guide its response to perform the task as shown in figure 13. This approach helps the model contextualize the task at hand, using the provided examples as a reference for the desired output or response style. It's especially beneficial for tasks where the context or format is more nuanced, and additional examples help the model understand the subtleties of the request [24]. This approach is particularly effective for tasks that require a nuanced understanding or for which the model may not have been explicitly trained. The examples act as a reference point, helping the model to generate responses that are aligned with the style, format, or content of the given examples.

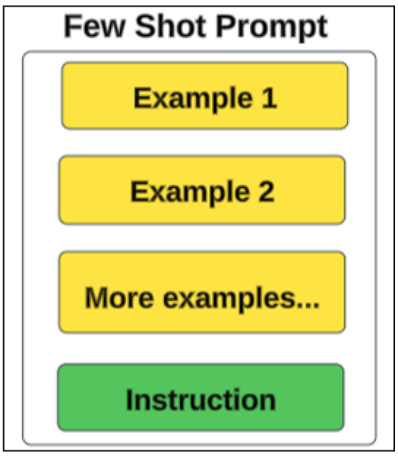

Figure 12: Few Shot Prompt

Few - shot prompting proves highly effective for tasks such as classification and prioritization. Figure [13] illustrates this point: by presenting the model with a handful of examples that demonstrate the classification and prioritization of bugs based on factors like severity and complexity, the model learns to accurately categorize and prioritize new bug reports. These examples serve as learning templates, enabling the model to apply similar reasoning to novel reports, thus demonstrating the practical utility of few - shot prompting in complex decision - making scenarios.

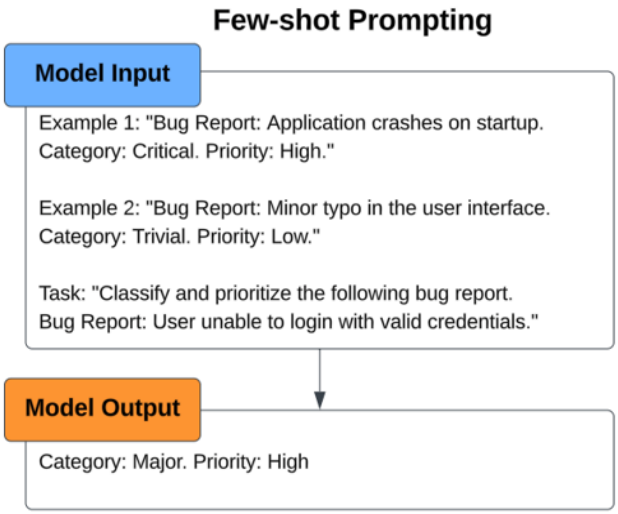

Figure 13: Few Shot Prompt Example

While few - shot prompting offers greater flexibility than zero - shot prompting, it necessitates the careful selection of examples. The efficacy of this method heavily relies on the quality and diversity of the examples provided. Examples that are not representative or overly varied may hinder the model's ability to accurately discern the task's context. Although curating relevant examples for few - shot prompting can be time - consuming, it significantly boosts the model's accuracy and relevance for contextually complex tasks. Contrarily, zero - shot prompts, which lack examples, may sometimes outperform few - shot prompts, particularly in tasks within the model's strong knowledge domains. This underscores the necessity of understanding the model's capabilities and selecting the most suitable prompting technique for optimal performance.

4.6 LLM settings: Temperature and Top - p

In the realm of Large Language Models (LLMs), settings like temperature and top - p are instrumental in defining the nature of the model's responses. These parameters act as controls to fine - tune the model's creativity and precision.

4.6.1 Temperature: This parameter controls the randomness or predictability of the generated output. A lower temperature setting results in more deterministic and predictable responses. It's particularly useful when accuracy and precision are paramount, such as in technical or factual queries [25]. In contrast, a higher temperature setting increases randomness, leading to more creative and diverse outputs. This setting is beneficial for brainstorming sessions or when seeking innovative solutions.

4.6.2 Top - p (Nucleus Sampling): Top - p controls the breadth of the model's consideration set for the next word in a sequence. It essentially narrows or broadens the model's focus when generating responses [26]. A lower top - p value makes the model's responses more focused and conservative, while a higher value allows for a wider array of possibilities, introducing a level of unpredictability and variety in the responses.

4.6.3 Examples:

- Low Temperature Setting:
- Context: In an engineering context, where precision and adherence to standards are crucial, a low temperature setting ensures that the model's responses are accurate and reliable.
- Prompt: "List the key factors in choosing materials for high - temperature industrial applications. "
- Output: "Key factors include thermal stability, resistance to oxidation and corrosion, mechanical strength at high temperatures, and compatibility with the operating environment. "
- High Temperature Setting:
- Context: For creative tasks, such as conceptual design or ideation in engineering, a higher temperature setting can stimulate more innovative and varied responses.
- Prompt: "Suggest an innovative approach to harnessing wind energy in urban areas. "
- Output: "Consider designing compact, vertical - axis wind turbines integrated into the structure of skyscrapers, utilizing the wind channels created between buildings to generate energy. "

While adjusting these parameters, it's important to strike a balance between creativity and relevance, especially in professional fields like engineering. For instance, in safety - critical applications, a lower temperature ensures adherence to standards and accuracy. On the other hand, in design and ideation phases, a higher temperature can unlock creative potentials. Additionally, it's worth noting that some models or interfaces, like ChatGPT, may have fixed settings for these parameters, focusing on delivering balanced and contextually appropriate responses.

5. Conclusion

In conclusion, this paper has comprehensively explored the transformative role of prompt engineering in enhancing the efficacy of Large Language Models (LLMs), particularly in engineering applications. We have systematically unpacked various methodologies of prompt crafting, emphasizing their significance in optimizing LLM outputs for precision, efficiency, and contextual relevance. Our findings underline the importance of strategic prompt design in harnessing the full potential of LLMs, setting a foundation for future innovations in human - AI collaboration within the engineering domain. The continued evolution of prompt engineering promises to refine and elevate our interaction with AI, presenting exciting prospects for the advancement of AI in practical and theoretical contexts.

References

[1] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer - Smith, J. and Schmidt, D. C., 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv: 2302.11382.

[2] Kojima T, Gu SS, Reid M, Matsuo Y, Iwasawa Y. Large language models are zero - shot reasoners. Advances in Neural Information Processing Systems.2022; 35: 22199–22213.

[3] Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., He, H., Li, A., He, M., Liu, Z. and Wu, Z., 2023. Summary of chatgpt - related research and perspective towards the future of large language models. Meta - Radiology, p.100017.

[4] Chen, B., Zhang, Z., Langrené, N. and Zhu, S., 2023. Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review. arXiv preprint arXiv: 2310.14735.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems, 30.

[6] Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E. and Brynjolfsson, E., 2021. On the opportunities and risks of foundation models. arXiv preprint arXiv: 2108.07258.

[7] Devlin, J., Chang, M. W., Lee, K. and Toutanova, K., 2018. Bert: Pre - training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 1810.04805.

[8] OpenAI. GPT - 4 Technical Report; 2023. ArXiv: 2303.08774.

[9] Anthropic.: Claude2. Accessed: 2023 - 7 - 27. figshare https: //www.anthropic. com.

[10] Sarkhel, R., Huang, B., Lockard, C. and Shiralkar, P., 2022. Label - Efficient Self - Training for Attribute Extraction from Semi - Structured Web Documents. arXiv preprint arXiv: 2208.13086.

[11] Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R. and McHardy, R., 2023. Challenges and applications of large language models. arXiv preprint arXiv: 2307.10169.

[12] Baidoo - Anu D, Ansah LO. Education in the era of generative artificial intelligence (AI): understanding the potential benefits of ChatGPT in promoting teaching and learning. Journal of AI.2023; 7 (1): 52–62.

[13] Lu Y, Bartolo M, Moore A, Riedel S, Stenetorp P. Fantastically ordered prompts and where to find them: overcoming few - shot prompt order sensitivity. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics; 2022. p.8086–8098.

[14] Webson A, Pavlick E. Do prompt - based models really understand the meaning of their prompts? In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies; 2022. p.2300–2344.

[15] Shanahan M, McDonnell K, Reynolds L. Role - play with large language models; 2023. ArXiv: 2305.16367.

[16] Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad.2021. A comprehensive survey on word representation models: From classical to state - of - the - art word representation language models. ACM Trans. Asian Low Resour. Lang. Inf. Process.20, 5 (2021)

[17] Radford, A., Karthik, D., Christian, S., Beechan, M., Jones, L.,. . . & Marris, M. (2019). "Language models are unsupervised multi task learners. " OpenAI Blog.

[18] Xu X, Tao C, Shen T, Xu C, Xu H, Long G, et al. Re - reading improves reasoning in language models; 2023. ArXiv: 2309.06275.

[19] Luo L, Ao X, Song Y, Li J, Yang X, He Q, et al. Unsupervised neural aspect extraction with sememes. In: IJCAI; 2019. p.5123–5129.

[20] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V. and Zhou, D., 2022. Chain - of - thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems, 35, pp.24824 - 24837

[21] Zhang Z, Gao J, Dhaliwal RS, Jia - Jun Li T. VISAR: a human - AI argumentative writing assistant with visual programming and rapid draft prototyping; 2023. ArXiv: 2304.07810.

[22] Shyr C, Hu Y, Harris PA, Xu H. Identifying and extracting rare disease phenotypes with Large language models; 2023. ArXiv: 2306.12656

[23] Gerardo Adesso. Towards The Ultimate Brain: Exploring Scientific Discovery with ChatGPT AI . Authorea. February 21, 2023.

[24] Reynolds L, McDonnell K. Prompt programming for large language models: beyond the few - shot paradigm. In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems; 2021. p.1 - 7

[25] Ficler J, Goldberg Y. Controlling linguistic style aspects in neural language generation. In: Proceedings of the Workshop on Stylistic Variation; 2017. p.19 94–104.
[26] Holtzman A, Buys J, Du L, Forbes M, Choi Y. The curious case of neural text degeneration. In: International Conference on Learning Representations; 2020.