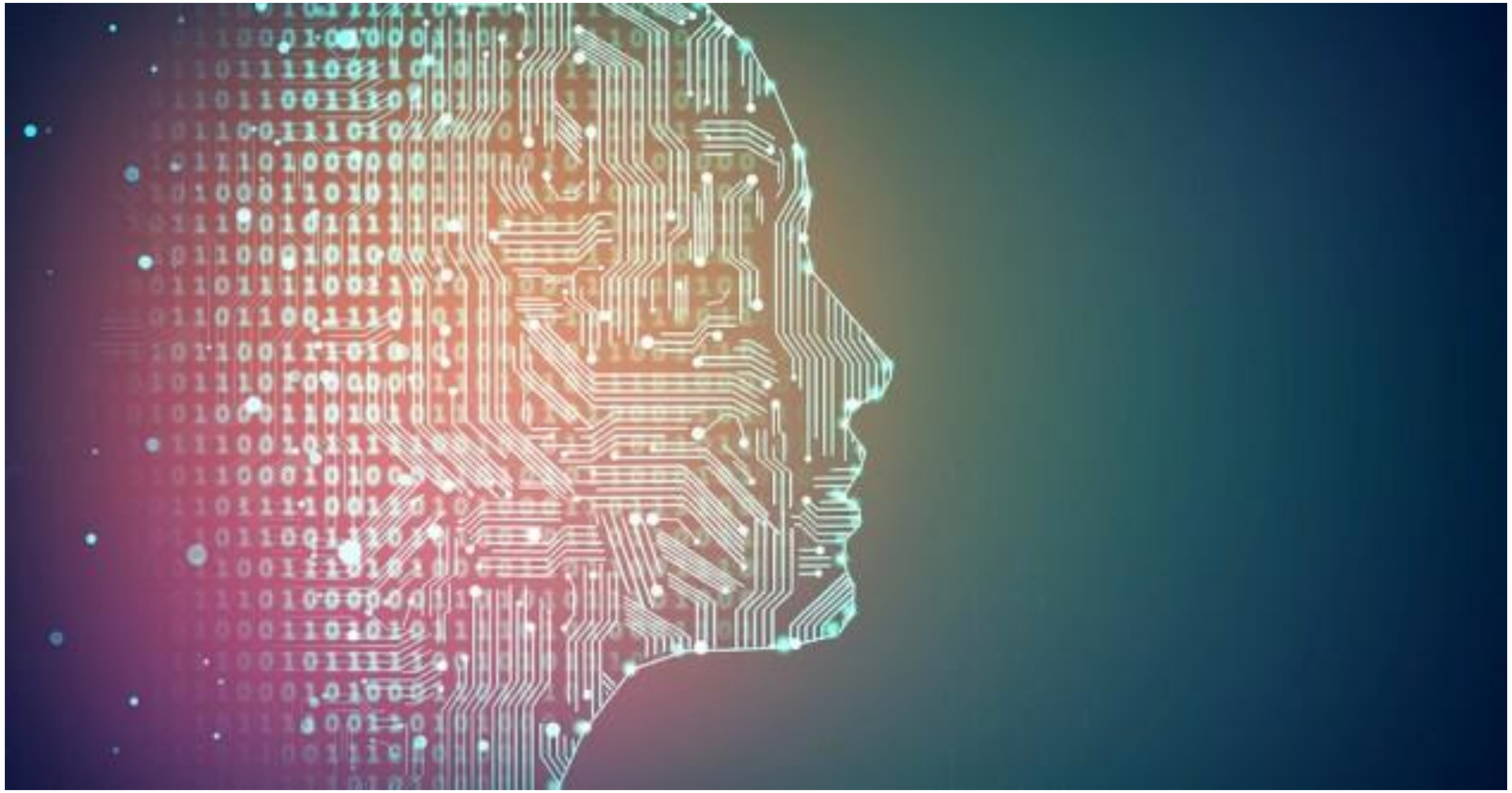
Generative AI & Large Language Models

Prompt Engineering – PART I

Thuan L Nguyen, Ph.D.

2: Generative AI & LLMs: Prompt Engineering



AI Deep learning (Source: mindovermachines.com)

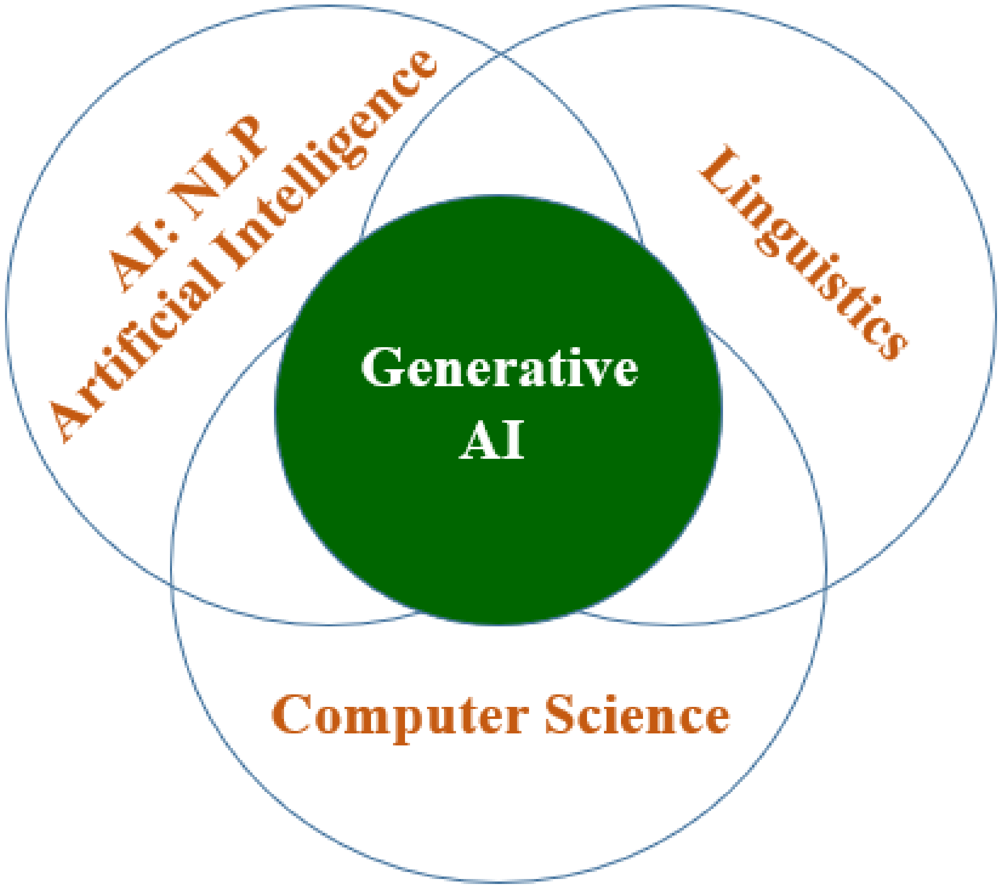
3: Generative AI & LLMs: Prompt Engineering

1. Generative AI: LLM: Prompt Engineering: Introduction
2. Generative AI: LLM: Prompt Engineering: Definition
3. Generative AI: LLM: Prompt Engineering: Types: Zero-Shot, Few-Shot
4. Generative AI: LLM: Prompt Engineering: Impacts: Benefits & Pitfalls
5. Generative AI: LLM: Prompt Engineering: Designing Prompts: Overview
6. Generative AI: LLM: Prompt Engineering: Designing Prompts: Structures & Elements
7. Generative AI: LLM: Prompt Engineering: Designing Prompts: Step by Step

4: Generative AI & LLMs: Prompt Engineering

Generative AI & Large Language Models

Foundational Sciences & Technologies



Generative AI is based on the NLP technologies such as Natural Language Understanding (NLU) and Conversational AI (AI Dialogues) - Those among the most challenging tasks AI needs to solve.

5: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Overview

- The advent of large language models (LLMs) like GPT, Gemini, and others has revolutionized the way humans interact with artificial intelligence (AI) neural networks.

- These models, trained on massive datasets of text and code, possess remarkable abilities to generate human-quality text, translate languages, write different kinds of creative content, and answer your questions in an informative way.

- However, harnessing the full potential of these powerful tools requires a nuanced understanding of how to effectively communicate with them. This is where prompt engineering comes into play.

6: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Definition

- The Prompt engineering is the art and science of crafting effective inputs, known as prompts, to guide a large language model (LLM) or other generative AI systems while the model to produce or generate responses
  - It involves carefully designing and structuring text-based instructions or queries to elicit desired responses
  - Essentially, it's about learning the language of AI and speaking it fluently to achieve your goals.
- Prompt engineering is not just about asking simple questions; it's a strategic approach that requires understanding the nuances of the model, its strengths and limitations, and the specific task at hand. It's a blend of creativity, experimentation, and iterative refinement.

6: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Definition

In other words:

- Prompt engineering can be defined as the discipline of designing and optimizing input prompts for AI models, particularly large language models, to elicit desired responses.

- It involves understanding the model's capabilities and limitations, experimenting with different phrasing and structures, and iteratively refining prompts to achieve the best possible results.

- It's a blend of art and science, requiring creativity, technical understanding, and a systematic approach.

7: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: Zero-Short Prompting

- Zero-shot prompting involves asking the LLM to perform a task without providing any explicit examples of how to complete that task. The model relies solely on its pre-trained knowledge and the instructions given in the prompt to generate a response.

- Example:
  - Prompt: "Translate the following English sentence into French: 'The cat sat on the mat.'"
  - Expected Output: "Le chat était assis sur le tapis.“

- Python code is not typically necessary for Zero-Shot prompting.

8: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: Few-Short Prompting

- Few-shot prompting provides the LLM with a few examples (typically 1-5) of the desired task before asking it to perform a similar task. These examples serve as a guide for the model to understand the expected format and style of the output.
- Example:
  - Prompt:
    - English: I love you.
    - French: Je t'aime.

    - English: Good morning.
    - French: Bonjour.

  - English: How are you?
  - French:
  - Expected Output: "Comment allez-vous?“
- Python code is not typically necessary for Few-Shot prompting.

9: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

- In-context learning is a prompting technique where, instead of explicitly instructing the LLM with rules or a detailed description of the desired task, the user provides a few examples of the task within the prompt itself.
- The LLM "learns" the task by observing the pattern in these examples and then applies that pattern to complete a new, related input.
  - It's a form of few-shot learning, but it's crucial to understand that the LLM doesn't update its model weights during this process.
  - It's not fine-tuning. The learning is purely based on the context provided in the prompt.
- It is like showing someone a few examples of how to solve a particular type of puzzle. The user doesn’t give the model the explicit rules, but by seeing the solved examples, the model can infer the underlying logic and apply it to a new puzzle of the same type.

10: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Key Characteristics of In-Context Learning (ICL):

- **No Parameter Updates**: The LLM's internal parameters remain unchanged. The "learning" happens solely through the context provided in the prompt.
- **Few-Shot Learning**: ICL often uses a small number of examples (usually 1-5, but can be more, limited by the prompt's maximum token length). The more relevant and diverse the examples, the better the performance.
- **Pattern Recognition**: The LLM identifies the relationship between the input and output in the provided examples. This relationship could be anything from a text transformation, sentiment analysis, translation, or code generation.
- **Prompt-Based**: The entire "learning" and execution of the task are contained within the prompt. This makes it very flexible and easy to adapt to new tasks without retraining.

11: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Key Characteristics of In-Context Learning (ICL):

- Context Window Limitation: The effectiveness of ICL is constrained by the LLM's **context window** (maximum prompt length). Too many examples, or examples that are too long, might exceed this limit.

12: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Key Characteristics of In-Context Learning (ICL):

- Context Window Limitation: The effectiveness of ICL is constrained by the LLM's **context window** (maximum prompt length). Too many examples, or examples that are too long, might exceed this limit.

Why is In-Context Learning Important?

- **Reduced Training Overhead**: It eliminates (or drastically reduces) the need for large, labeled datasets and computationally expensive fine-tuning.
- **Rapid Task Adaptation**: The user can quickly switch between tasks by simply changing the examples in the prompt.
- **Accessibility**: It makes powerful LLMs accessible to users without extensive machine learning expertise. The user does not need to be a data scientist to use it effectively.

13: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Examples of **In-Context Learning**

Example 1: Sentiment Classification

Imagine some user want to classify the sentiment of movie reviews as either "positive" or "negative". Here's how we can use ICL:

- Text: This movie was absolutely amazing! The acting was superb, and the plot was captivating.
- Sentiment: positive

- Text: I found the film to be quite boring. The storyline was predictable, and the characters were dull.
- Sentiment: negative

- Text: The special effects were incredible, and the soundtrack was phenomenal. I loved it!
- Sentiment: positive

- Text: The dialogue was clunky and unnatural. I couldn't connect with the story at all.
- Sentiment: negative

14: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Examples of **In-Context Learning**

Example 1: Sentiment Classification (Cont)

Imagine some user want to classify the sentiment of movie reviews as either "positive" or "negative".

- Text: The cinematography was stunning, a real treat to watch.
- Sentiment: ?

NOTES:

In this prompt, we provide four above examples of text-sentiment pairs. The LLM is expected to recognize the pattern and, based on this context, predict the sentiment of the final text ("The cinematography was stunning, a real treat to watch.") as "positive".

15: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Example 2: English to French Translation

Let's demonstrate a simple translation task:

- English: Hello, how are you?
- French: Bonjour, comment allez-vous?

- English: I like to eat apples.
- French: J'aime manger des pommes.

- English: Where is the library?
- French: Où est la bibliothèque?

- English: Good morning.
- French: ?
- Here, we're showing the LLM three English-to-French sentence pairs. The LLM should infer the translation task from these examples and then translate "Good morning" into French ("Bonjour").

16: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Example 2: English to French Translation (Cont.)

Let's demonstrate a simple translation task (Cont.):

- English: Good morning.
- French: ?

NOTES:
--) Here, we're showing the LLM three English-to-French sentence pairs. The LLM should infer the translation task from these examples and then translate "Good morning" into French ("Bonjour").

17: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Types: In-Context Learning: Prompting with Demonstration

Conclusion: The power of context.

- In-context learning is a paradigm shift in how we interact with LLMs. It leverages the impressive pattern recognition capabilities of these models to perform tasks without any explicit programming or fine-tuning.

- By carefully crafting prompts with relevant examples, the user can guide LLMs to achieve remarkable results across a wide range of applications. This makes generative AI more flexible, adaptable, and accessible to a broader range of users.

- It also demonstrates that a well-designed prompt, using the correct context, can often be as powerful as, and is often easier to implement and maintain than a finetuned model, especially when the tasks are varied.

18: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Impacts

Benefits

- **In-context Improved Performance**: Well-crafted prompts can significantly improve the accuracy, relevance, and creativity of LLM outputs.
- **Enhanced Control**: Prompt engineering allows users to steer the LLM's behavior and generate outputs that align with their specific needs.
- **Increased Efficiency**: By optimizing prompts, users can reduce the time and effort required to achieve desired results.
- **Democratization of AI**: Effective prompt engineering makes LLMs more accessible to users without deep technical expertise.

19: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Impacts

Pitfalls

- **In-context Prompt Sensitivity**: LLMs can be highly sensitive to even minor changes in the prompt, leading to unpredictable outputs. This requires careful and iterative prompt design.

- **Bias Amplification**: If the training data contains biases, poorly designed prompts can amplify these biases in the generated output, leading to unfair or discriminatory results.

- **Hallucinations**: LLMs can sometimes "hallucinate" information, generating outputs that are factually incorrect or nonsensical. Careful prompt engineering and output verification are essential to mitigate this.

20: Generative AI & LLMs: Prompt Engineering

Prompt Engineering: Impacts

Pitfalls (Cont.)

- **In-context Security Risks**: Maliciously crafted prompts can potentially be used to trick LLMs into generating harmful or inappropriate content. Robust safety mechanisms are crucial.

- **The "Black Box" Problem**: Understanding *why* a particular prompt works or doesn't work can be challenging, as the internal workings of LLMs are often opaque. This makes prompt engineering a somewhat empirical process.

- **Over-Optimization**: Focusing too much on optimizing for specific outputs can lead to overfitting the prompt, making it less effective for other tasks or datasets.