Prompt Engineering

Prompting Patterns and Chain-of-Thought

Ramaseshan Ramachandran

①

②

③

④

OVERVIEW OF GENERATIVE ARTIFICIAL INTELLIGENCE (AI)

- Subset of AI that uses machine learning to create new, original content.
- Generates images, text, or music by learning patterns from existing data.
- Large Language Models (LLMs) are a key type of generative AI.
- LLMs, like ChatGPT, produce natural language texts.
- Built with millions or billions of parameters, trained on vast text datasets.
- Capable of tasks: answering questions, summarizing, writing essays, and generating stories.
- Learn from their outputs, improving over time.
- Lack true knowledge or understanding of generated material.
- Generate content based on existing patterns, not comprehension.
- May present false facts as true ("hallucinations").

LLMs cannot read your mind

Prompt engineering is a rapidly developing field that influences the interactions and outputs of generative artificial intelligence models.

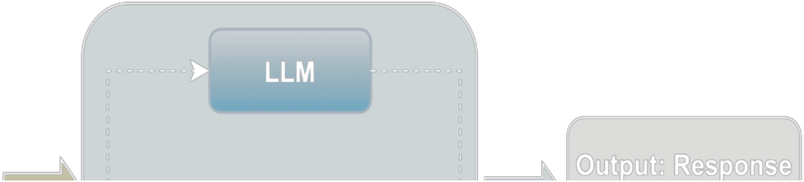
- Textual input to guide LLM model output.
- It could be a simple question to detailed descriptions or tasks
- Depends on the model, whether image generation (DALLůE 3, Midjourney, Stable Diffusion) or LLMs (GPT-4/5, Claude, Gemini, Llama, Grok, and others).
- Key components: Instructions, questions, input data, examples.

TYPES OF PROMPTS IN LLMS

- Basic Prompts: Direct questions or simple instructions.
- Advanced Prompts: Complex structures, logical reasoning.
- Chain of Thought Prompting: Guiding model through reasoning process.
- Contextual Prompts: Considering specific context or domain.
- Exemplar-based (few-shot) Prompts: Providing examples for guidance.

The less the model guesses, the more likely it is to get what you want.

COMPONENTS OF PROMPT ENGINEERING[**sahoo2024systematicsurveypromptengineering**]



- Prompt-based learning reframes traditional NLP tasks by leveraging LLMS trained to predict text probabilities.
- It transforms tasks into a text completion or fill-in-the-blank format.
- Instead of predicting $p(y \mid x; \theta)$ (as in supervised learning), it focuses on modeling $p(x; \theta)$, where x is the input text.
- This reduces or eliminates the need for large annotated datasets.

KEY IDEA AND MATHEMATICAL OBJECTIVE

Key Idea:

- Use a pre-trained LLM to evaluate the probability of text completions.
- Leverage LLM's knowledge acquired from pre-training on large corpora.

Mathematical Objective:

- Given an input x, find the output $\hat{y}$ that maximizes:

$$\hat{y} = \arg\max_{y} P(p(x, y); \theta)$$

- Here:
  - $p(x, y)$: A prompt combining input x and possible output y.
  - θ: Parameters of the language model.

PROMPT CONSTRUCTION

What is a Prompt?

- A prompt $p(x, y)$ is a reformulation of input x that guides the LLM to predict y.
- Includes:
  - Input slot $[X]$ for x.
  - Answer slot $[Z]$ for z, the intermediate text leading to y.

Example: Sentiment Analysis

- Input Text (x): I love this movie.
- Template: [X] Overall, it was a [Z] movie.
- Prompt ($p(x, y)$): I love this movie. Overall, it was a [Z] movie.

INPUT SLOT ([X])

- Represents the location in the template where the input x is inserted.
- Example:
  - Input: $x =$ I love this movie.
  - Slot [X] is replaced with this text.
  - Template: [X] Overall, it was a [Z] movie.

ANSWER SLOT ([Z])

- Placeholder for the intermediate reasoning or output z.
- z is an intermediate text representation (e.g., descriptive reasoning or classification) that leads to the final prediction y.
- Example:
  - In sentiment analysis, z could be positive.
  - Template: [X] Overall, it was a [Z] movie.

TEMPLATE

- A predefined structure that incorporates [X] and [Z].
- Example: Template: [X] Overall, it was a [Z] movie.
- Slots [X] and [Z] allow flexibility in adapting the input and reasoning process.

FINAL PROMPT $p(x, y)$

- The completed prompt is formed by filling the slots [X] and optionally predicting [Z].
- Example:
  - Input: $x =$ I love this movie.
  - Template: [X] Overall, it was a [Z] movie.
  - Final Prompt: $p(x, y) =$ I love this movie. Overall, it was a [Z] movie..

SCORING THE PROMPT

- Evaluate the probability assigned by the LLM to the prompt $p(x, y)$: $P(p(x, y); \theta)$
- Key elements:
  - $p(x, y)$: The prompt containing the input x and candidate output y.
  - θ: Parameters of the language model.
- Higher probabilities indicate more plausible completions.

SELECTING THE OUTPUT

- Choose $\hat{y}$ as the y that maximizes $P(p(x,y);\theta)$:

$$\hat{y} = \arg\max_{y} P(p(x,y);\theta)$$

- Process:
  1. Evaluate $P(p(x,y);\theta)$ for all possible y.
  2. Select y with the highest probability.
- $\hat{y}$: The predicted or selected output.

EXAMPLE: SENTIMENT ANALYSIS

- **Input**: $x =$ I love this movie.
- **Template**: [X] Overall, it was a [Z] movie.
- **Scoring**:
  - Possible completions for [Z]:
    - $z = \text{great}, P(p(x,y);\theta) = 0.8$
    - $z = \text{bad}, P(p(x,y);\theta) = 0.2$
  - *Prompts*:
    - $p(x,y) =$ I love this movie. Overall, it was a great movie.
    - $p(x,y) =$ I love this movie. Overall, it was a bad movie.
- **Selection**:
  - Select z with the highest probability: $z = \text{great}$.
  - Final output $\hat{y}$: Positive sentiment.

SIGNIFICANCE OF SCORING THE PROMPT

- Ensures the model predicts the most plausible and contextually appropriate output.
- Combines:
  - Effective prompt design.
  - The LLM's generative capabilities.
- Balances probability-based evaluation and structured reasoning.

ADVANTAGES OF PROMPT-BASED LEARNING

- Requires fewer labeled datasets compared to supervised learning.
- Leverages the general knowledge encoded in pre-trained LLMs.
- Adapts easily to new tasks by simply designing appropriate prompts.
- Can incorporate task-specific examples through few-shot learning.

CHALLENGES OF PROMPT-BASED LEARNING

- Designing effective prompts is non-trivial.
- Performance is sensitive to minor changes in prompt phrasing.
- Might require manual effort or automated prompt engineering.
- May struggle with tasks requiring highly specific domain knowledge.

PROMPT ADDITION

Step Description: A prompting function $f_{\text{prompt}}(\cdot)$ is applied to modify the input text x into a prompt x' as follows:

$$x' = f_{\text{prompt}}(x)$$

Key Steps in f_{prompt}:

1. **Apply a Template:**
   - A textual string with two slots:
     - $[X]$: Input slot for x.
     - $[Z]$: Answer slot for an intermediate answer text z, which will later map to y.
2. **Fill the Slots:**
   - Fill $[X]$ with the input text x.

EXAMPLES OF PROMPT ADDITION

Example 1: Sentiment Analysis

- Input Text (x): I love this movie.
- Template: [X] Overall, it was a [Z] movie.
- Prompt (x'): I love this movie. Overall, it was a [Z] movie.

Example 2: Machine Translation

- Input Text (x): A sentence in Finnish.
- Template: Finnish: [X] English: [Z]
- Prompt (x'): Finnish: (input sentence) English: [Z]

ADVANTAGES OF PROMPT-BASED LEARNING

- Bypasses the need for large labeled datasets.
- Leverages the LM's general knowledge (pre-trained on large corpora).
- Can adapt to new tasks with minimal data.

CHALLENGES AND EXTENSIONS

Challenges:

- Designing effective prompts (manual or automated).
- Prompt sensitivity: Small changes in phrasing can affect results.

Extensions:

- Few-shot prompting: Use a few labeled examples in the prompt.
- Chain-of-thought prompting: Guide the LM with step-by-step reasoning.

CHAIN OF THOUGHTS[**wei2023chainofthoughtpromptingelicitsreasoning**]

- **Objective:** Explore how generating a chain of thought improves large language models' reasoning abilities.
- **Method:** Utilize *chain-of-thought prompting*. It provides intermediate reasoning steps as exemplars to enhance performance on complex tasks.
- **Key Findings:** Significant improvements in arithmetic, commonsense, and symbolic reasoning tasks.

ZERO-SHOT PROMPTING

- **Zero-shot prompting** refers to performing a task using a language model (LM) without any task-specific labeled examples or additional training.
- The LM relies on its pre-trained knowledge to interpret and complete the task based on the provided instructions in the prompt.
- No prior task-specific fine-tuning or examples are provided.
- Works best when the task is clear and the LM has encountered similar contexts during pre-training.

KEY IDEA OF ZERO-SHOT PROMPTING

Objective: Use the LM's pre-trained knowledge to generate the desired output y for an input x, guided solely by the prompt.

- Given an input x, construct a natural language instruction in the prompt.
- The LM predicts the output y directly:

$$\hat{y} = \arg\max_{y} P(p(x, y); \theta)$$

- Example structure of the prompt:
  - Task description: "Translate the following text."
  - Input: "[Input text]."
  - Instruction guides the LM to generate the output.

EXAMPLE 1: SENTIMENT ANALYSIS

Task: Determine the sentiment (positive, negative, or neutral) of the input text.
Prompt:

- "What is the sentiment of the following review? 'I love this product!'"

LM Response:

- "Positive."

Key Point:

- The LM interprets the instruction and predicts the sentiment based on its pre-trained knowledge.

EXAMPLE 2: MACHINE TRANSLATION

Task: Translate a sentence from English to French. **Prompt:**

- "Translate the following sentence from English to French: 'I am going to the market.'"

LM Response:

- "Je vais au marché."

Key Point:

- The LM uses its pre-trained multilingual knowledge to perform the task without needing any examples.

EXAMPLE 3: TEXT COMPLETION

Task: Complete a sentence based on the given context. **Prompt:**

- "Complete the following sentence: 'The quick brown fox jumps over the...'"

LM Response:

- "lazy dog."

Key Point:

- The LM draws on commonly seen patterns to generate an appropriate and meaningful completion.

ADVANTAGES OF ZERO-SHOT PROMPTING

- No labeled data or fine-tuning required for specific tasks.
- Efficient and flexible for a wide range of tasks.
- Allows the LM to leverage its general knowledge from pre-training.
- Easy to use with clear instructions in natural language.

CHALLENGES OF ZERO-SHOT PROMPTING

- Performance depends heavily on the clarity and phrasing of the prompt.
- May struggle with complex or niche tasks outside the LM's pre-trained knowledge.
- No feedback or task-specific learning occurs.
- Sensitive to ambiguous or poorly designed prompts.

SUMMARY OF ZERO-SHOT PROMPTING

- Zero-shot prompting uses a language model's pre-trained knowledge to perform tasks without labeled data or examples.
- It reframes tasks using natural language instructions within the prompt.
- Works best for tasks where the LM has sufficient pre-trained knowledge.
- Clear prompt design is crucial for effective zero-shot prompting.

CHAIN-OF-THOUGHT PROMPTING

- **Definition:** A prompting method where models are given input-output pairs along with intermediate reasoning steps.
- **Benefits:**
  - Decomposes complex problems into manageable steps.
  - Provides interpretability and debugging opportunities.
  - Applicable to various reasoning tasks (e.g., math word problems).

CHAIN-OF-THOUGHT PROMPTING

- Chain-of-Thought (CoT) prompting is a technique that enables language models to perform complex reasoning tasks by breaking them into smaller, interpretable steps.
- Encourages the model to "think aloud" and generate intermediate reasoning steps instead of producing a single answer directly.
- Particularly effective for mathematical reasoning, logical inference, and commonsense reasoning.

KEY IDEA OF CHAIN-OF-THOUGHT PROMPTING

- Traditional prompting predicts output y directly:

$$\hat{y} = \arg\max_{y} P(y \mid x; \theta)$$

- CoT prompting predicts a sequence of intermediate reasoning steps:

$$\hat{z} = \arg\max_{z} P(z \mid x; \theta)$$

  where z represents reasoning steps, and the final answer $\hat{y}$ is derived from z.
- This step-by-step reasoning improves the model's ability to solve complex tasks.

EXAMPLE 1: MATHEMATICAL REASONING

Task: What is 17 multiplied by 23? **Prompt:**

- "Solve step-by-step: What is 17 multiplied by 23?"

CoT Reasoning:

- Step 1: "First, multiply 17 by 20 to get 340."
- Step 2: "Next, multiply 17 by 3 to get 51."
- Step 3: "Add the results: $340 + 51 = 391$."

Final Answer: 391.

EXAMPLE 2: COMMONSENSE REASONING

Task: If Alice has 3 apples and gives 2 to Bob, how many does she have left?
Prompt:

- "Think step-by-step: If Alice has 3 apples and gives 2 to Bob, how many does she have left?"

CoT Reasoning:

- Step 1: "Alice starts with 3 apples."
- Step 2: "She gives 2 apples to Bob, so she loses 2 apples."
- Step 3: "3 - 2 = 1."

Final Answer: 1 apple.

EXAMPLE 3: LOGICAL INFERENCE

Task: All cats are animals. Whiskers is a cat. Is Whiskers an animal? **Prompt:**

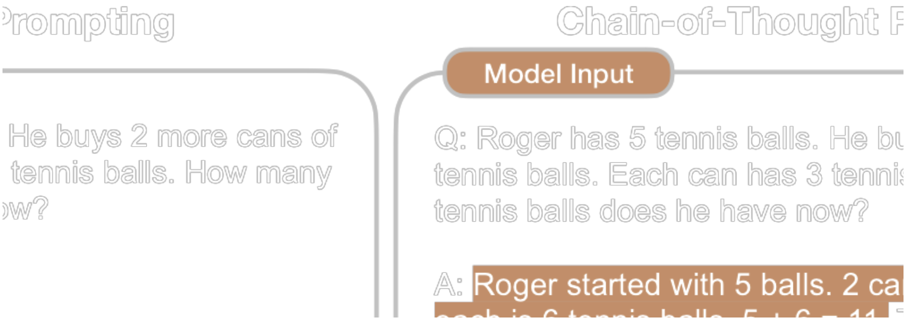
- "Reason step-by-step: All cats are animals. Whiskers is a cat. Is Whiskers an animal?"

CoT Reasoning:

- Step 1: "All cats are animals, so any cat must also be an animal."
- Step 2: "Whiskers is a cat."
- Step 3: "Therefore, Whiskers is an animal."

Final Answer: Yes, Whiskers is an animal.

STANDARD PROMPTS VS COT PROMPTS



ADVANTAGES OF CHAIN-OF-THOUGHT PROMPTING

- Improves performance on complex reasoning tasks.
- Makes the reasoning process interpretable and transparent.
- Reduces errors caused by directly predicting the output without intermediate steps.
- Generalizes well to tasks requiring multiple reasoning steps.

CHALLENGES OF CHAIN-OF-THOUGHT PROMPTING

- Sensitive to prompt design; poorly phrased prompts can lead to incorrect reasoning.
- May generate overly verbose or redundant reasoning steps.
- Requires larger or more capable language models for effective performance.
- Performance can degrade for tasks requiring domain-specific expertise.

SUMMARY OF CHAIN-OF-THOUGHT PROMPTING

- Chain-of-Thought (CoT) prompting enables models to solve complex tasks by reasoning step-by-step.
- Converts the problem into a sequence of intermediate reasoning steps before arriving at the final answer.
- Effective for mathematical, logical, and commonsense reasoning tasks.
- Clear, structured prompts are crucial for leveraging the benefits of CoT prompting.

BEYOND BASIC PROMPTING

- **Few-shot prompting**: include a handful of worked examples in the prompt
- **Self-consistency**: sample several chain-of-thought answers and take a majority vote
- **Retrieval-Augmented Generation (RAG)**: retrieve relevant documents and add them to the prompt to ground answers and reduce hallucination
- **Tool / function calling**: let the model invoke external tools (search, code, calculators, APIs)
- **ReAct**: interleave reasoning steps with tool actions
- **System prompts**: set persistent role, tone, and constraints separate from the user turn
- **Structured output**: request JSON or a schema for machine-readable results

BEST PRACTICES FOR CRAFTING PROMPTS

- Clear and concise language.
- Specific goals and requirements.
- Relevant context and examples.
- Iterative refinement and testing.
- Consider model capabilities and limitations.

CONCLUSION

- Prompts are crucial for effective AI output.
- Understanding prompt types and components is key.
- Crafting effective prompts requires clarity and specificity.
- Continuous refinement and testing ensure optimal results.
- Prompts will continue to evolve with AI advancements.

REFERENCES I