Learn and Grow

Generative AI LangChain

ASHIMA MALIK

ashimamalik58@gmail.com

PROMPTING 

- Prompt engineering is the art of crafting the text instructions that guide a large language model (LLM) towards generating the desired output.
- It's like providing clear and concise instructions to a powerful tool to ensure it delivers the best possible results.

How it Works:

Understanding the LLM: The first step is to understand the capabilities and limitations of the specific LLM you're using. Different LLMs might have different strengths and weaknesses, so tailoring prompts accordingly is crucial.

Formulating the Prompt: The prompt itself can take various forms, including questions, instructions, or creative writing prompts. The key is to be clear, concise, and provide all the necessary context for the LLM to understand your request.

Examples and Reformulations: Sometimes, providing the LLM with examples of desired outputs or reformulating your prompt in different ways can improve the results. This helps the LLM grasp the style and tone you're aiming for.

Benefits of Prompt Engineering:

Improved Accuracy and Relevance: Well-crafted prompts can significantly increase the accuracy and relevance of the LLM's outputs.

Unlocking Potential: Effective prompt engineering can unlock the full potential of LLMs, allowing them to perform tasks or generate creative text formats that might not be readily apparent without proper guidance.

Increased Efficiency: By guiding the LLM towards the desired outcome, prompt engineering can save time and resources compared to trial-and-error approaches.

ASHIMA MALIK

Top k

Top-K limits the next word prediction to the top K most probable words from the LLM's vocabulary, based on the current context.

It enforces a higher degree of control over the generated text by excluding less likely words. This can lead to more predictable and coherent outputs, but also potentially limit creativity and diversity.

Imagine the LLM is writing a sentence and predicts "The cat sat on the..." With Top-K=3, it might only consider the three most likely words to follow "the," like "mat," "rug," or "floor."

mat (0.4)

rug (0.3)

floor (0.2)

cloud (0.05)

The word/token is selected randomly but only from the top k words/tokens. For example, if k = 3, then, one of the words from the top probability words mat, rug, floor will be selected.

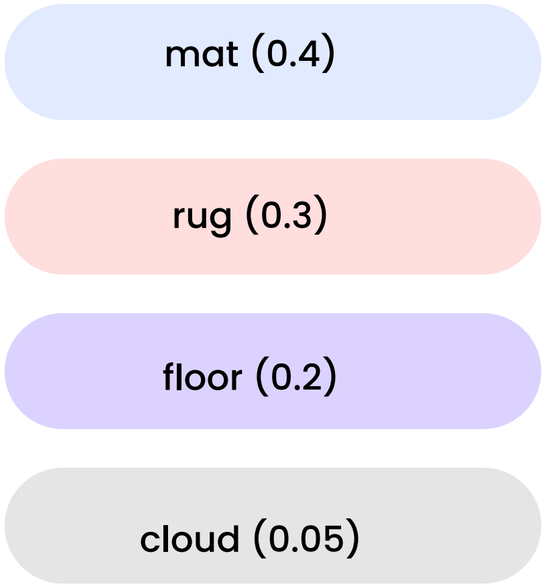
It enforces a higher degree of control over the generated text by excluding less likely words. This can lead to more predictable and coherent outputs, but also potentially limit creativity and diversity.

Top P

Top-P is a technique used in large language models (LLMs) to control the randomness and diversity of the generated text by focusing on the cumulative probability of the chosen words.

Its value is between 0 and 1.

Imagine the LLM is writing a sentence and predicts "The cat sat on the..."



1. The LLM considers these probabilities cumulatively. It starts adding the probabilities together:
    - Mat (0.4)

2. The LLM keeps adding probabilities until the cumulative sum reaches a pre-defined threshold (P). This threshold (P) is a value between 0 and 1 that you set. In this example, let's say the threshold (P) is set to 0.7.
Mat (0.4) - Doesn't reach the threshold yet.

3. The LLM keeps including words as long as the sum stays below the threshold. Here, it adds "cloudy":
    - Mat (0.4) + Rug (0.3) = 0.7 (Reached the threshold)

4. Once the threshold is reached (or exceeded), the LLM stops adding words and chooses a random word from the selected set. In this case, the chosen words are "rug" and "floor" because their combined probability reached the threshold. The LLM would then randomly pick one of these two words to proceed with the sentence generation.

- Higher Top-P value (closer to 1): Leads to more diverse and interesting outputs because the LLM considers a wider range of words even if their probabilities are lower. However, it might also introduce unexpected turns in the text.
- Lower Top-P value (closer to 0): Leads to more predictable and coherent outputs by focusing on the most probable words. However, it can limit creativity and diversity.

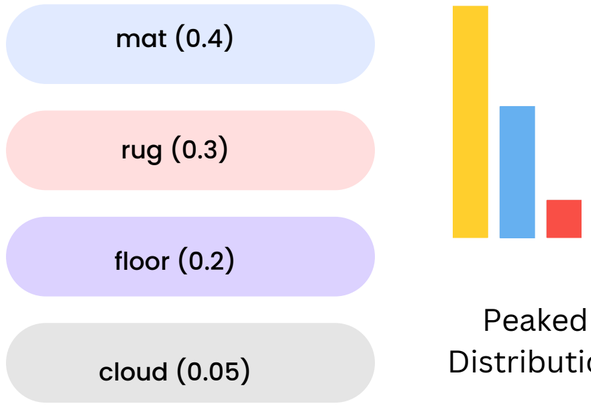
Benefits: Controls randomness and diversity, More natural-sounding outputs

ASHIMA MALIK

Temperature

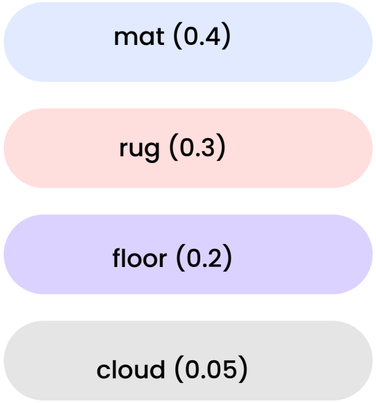
Temperature is a parameter that influences the randomness of the words chosen by the LLM during text generation.

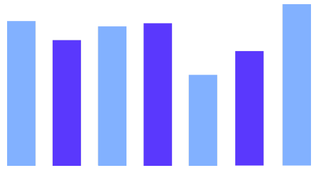
Imagine the LLM is writing a sentence and predicts "The cat sat on the..."



Peaked Distribution

Low Temperature (0 < T < 1): The adjusted probabilities become more concentrated on the highest probability word ("mat"), making the LLM more likely to choose the safest and most predictable option.





Flatter Probability Distribution

High Temperature (T > 1): The adjusted probabilities become more spread out, making the LLM more likely to choose less probable words like "rug" or even "cloud" for a more creative but potentially less relevant continuation.

Temperature of 1 (T = 1): This represents the original probabilities without any modification. The LLM will be chosen based on the inherent probability distribution.

Benefits of Temperature:

- Controls Creativity: Allows you to fine-tune the level of creativity and surprise in the generated text.
- Exploration vs. Certainty: High temperature encourages exploration of less likely but potentially interesting options. Low temperature prioritizes certainty and chooses the most probable continuation.

ASHIMA MALIK

Maximum Length

It is a setting that controls the total number of tokens the model can generate in its output. It essentially sets a limit on the length of the text the LLM will produce.

- Prevents overly long or irrelevant responses: Without a limit, LLMs might continue generating text even when it becomes redundant or strays from the original prompt or question.
- Manages computational cost: Generating text requires processing power, and longer outputs demand more resources. Setting a maximum length can help control the computational cost, especially when dealing with large datasets or limited computing power.
- Improves focus and coherence: By limiting the LLM's output length, you can encourage it to focus on the most relevant information and generate more concise and coherent responses.

Example:

- You set a maximum length of 100 tokens for the LLM to summarize a news article.
- The LLM analyzes the article and generates a summary that captures the key points within the 100-token limit.
- If the article is very long and the summary requires more than 100 tokens, the LLM will stop its output at the 100th token, potentially leaving out some details.

Token - smallest unit of text - word, punctuation mark or special character

Additional points to consider:

- The ideal maximum length depends on the specific task and desired output format. For short summaries, a lower limit might be appropriate, while creative text formats like poems or scripts could benefit from a higher limit.
- The maximum length setting is often used in conjunction with other parameters like temperature and top-k sampling to influence the content and style of the generated text.
- Some LLMs might allow specifying separate limits for the user prompt and the generated response within the overall maximum length.

ASHIMA MALIK

Frequency Penalty

Frequency penalty is a technique used in large language models (LLMs) to discourage repetitive and predictable outputs. It penalizes the LLM for generating tokens (words, punctuation marks, etc.) that have already appeared frequently in the current prompt and response.

LLM predicts the probability of each possible token based on the current context	Imagine the LLM is generating a sentence and has already used the word "the" twice.
penalty score is applied to each predicted token based on its frequency	The more a token has appeared already, the higher the penalty score it receives.
LLM adjusts the original probabilities by incorporating the penalty scores	This effectively reduces the probability of the LLM choosing frequently used tokens.
LLM samples a token based on the adjusted probabilities	This encourages the LLM to explore a wider range of vocabulary and generate more diverse and interesting outputs.

Benefits: Reduces repetitiveness, Improves fluency, Encourages creativity

Drawbacks: Might affect accuracy, Finding the right balance

ASHIMA MALIK

Presence Penalty

Presence penalty discourages the LLM from using words that have already appeared even once in the current context . A token that appears twice and a token that appears 10 times are penalized the same.

LLM predicts the probability of each possible token based on the current context

Imagine the LLM is generating a sentence and has already used the word "the" twice.

penalty score is applied to each predicted token based on its frequency

If the token is identical to a word already used in the prompt or response, it receives a penalty score.

LLM adjusts the original probabilities by incorporating the penalty scores

Words that have already been used have their probabilities reduced, making them less likely to be chosen again.

LLM samples a token based on the adjusted probabilities

This encourages the LLM to explore a wider vocabulary and avoid repetition.

Frequency Penalty vs. Presence Penalty:

- Frequency Penalty: Penalizes tokens based on how many times they have appeared in total (including the prompt and response).
- Presence Penalty: Penalizes tokens simply for appearing once before in the prompt or response.

LLM Settings

Setting	Function	How it Works	Affects Output	Use Case
Temperature	Controls randomness/creativity	Higher = More creative, less predictable, Lower = More conservative, predictable	Surprise, Uniqueness	Creative text formats (poems, code)
Top K	Limits next word options	Lower = Fewer options, more focused, Higher = More options, diverse but potentially less relevant	Focus, Determinism	Code generation (avoiding unexpected choices), Focused summaries
Top P	Filters next words based on cumulative probability	Lower = Only highly probable continuations, Higher = More options with probability exceeding threshold	Relevance, Focus vs. Diversity	Factual summaries (ensuring key points), Balancing interesting details with accuracy
Maximum Length	Limits output length	Sets a limit on the number of words/tokens	Conciseness	Summaries, Code snippets
Repetition Penalty	Discourages repetitive words/phrases	Assigns penalty for high repetition, encourages varied text	Fluency, Originality	All text formats (avoiding monotony)
Presence Penalty (Less Common)	Discourages specific unwanted words/phrases	Penalizes outputs with specific words/phrases	Content Control	Avoiding offensive language, Specific domain restrictions

Elements of Prompts

1. Instruction/Task:

This is the core of your prompt. It clearly states what you want the LLM to do.
Be specific! Instead of a vague prompt like "Write something interesting," instruct the LLM to "Write a poem about a robot who falls in love with a human."

2. Context (Optional):

Provide the LLM with relevant background information or details about the situation.
This helps the LLM understand the prompt better and generate more accurate and relevant outputs.
Example: "The year is 2042. Robots are commonplace..." (context for the robot love poem).

3. Examples (Optional):

Sometimes, providing the LLM with examples of desired outputs can be helpful, especially for creative tasks.
This gives the LLM a better idea of the style and tone you're aiming for.
Example: You could provide a few lines of poetry as an example for the robot love poem prompt.

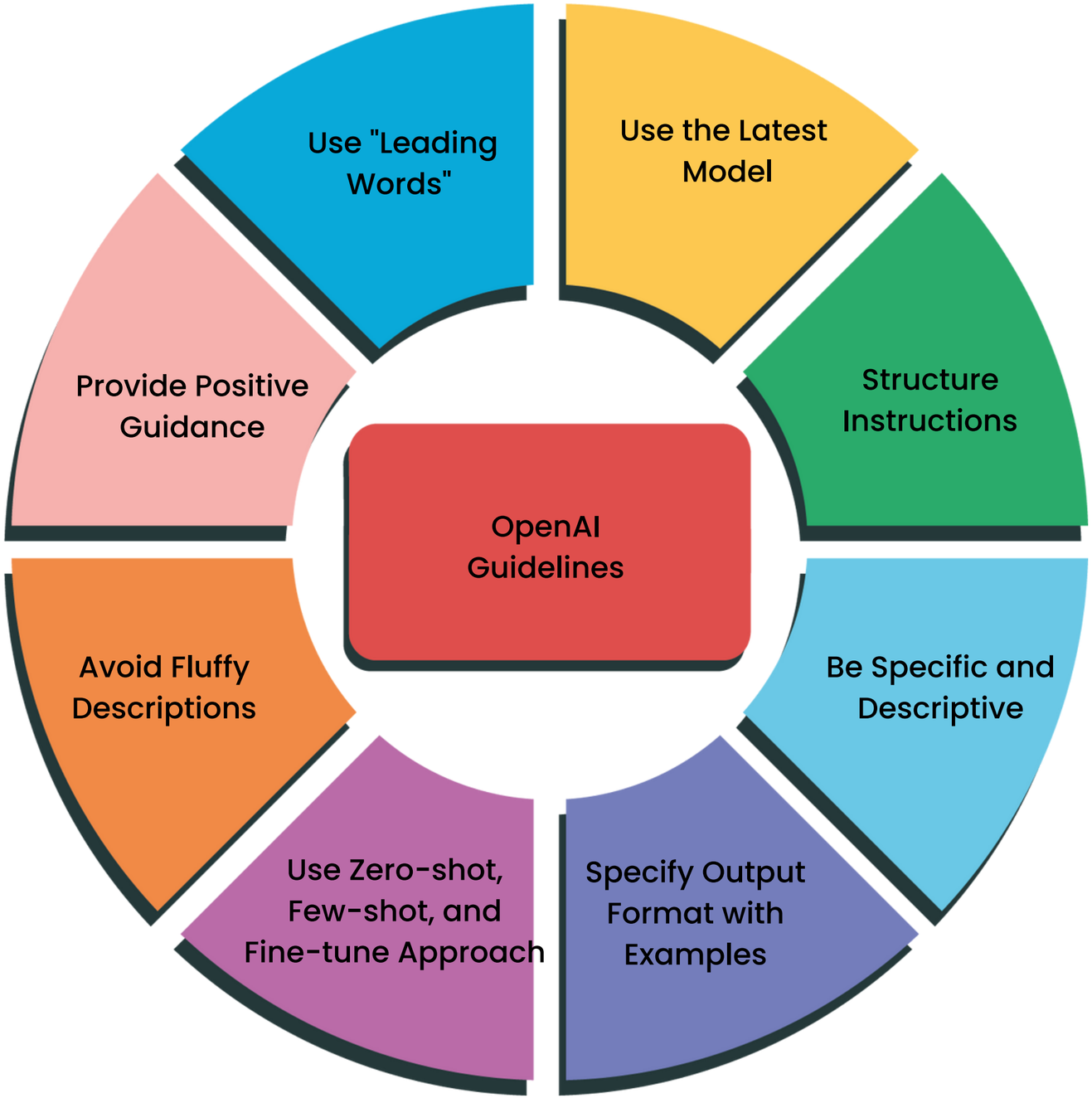
4. Persona (Optional):

In some cases, you might want to specify the persona or voice the LLM should use in the output.
This could be a specific character, a narrator, or even a particular writing style (e.g., formal, informal).
Example: "The poem should be narrated from the robot's perspective, full of longing and wonder."

5. Format (Optional):

Depending on the task, you might specify the desired output format.
This could be a specific number of words, a particular style of writing (e.g., code, script), or a creative text format (e.g., poem, email).
Example: "I want the poem to be a sonnet (14 lines with a specific rhyme scheme)."

ASHIMA MALIK



ASHIMA MALIK

Zero-Shot Prompting

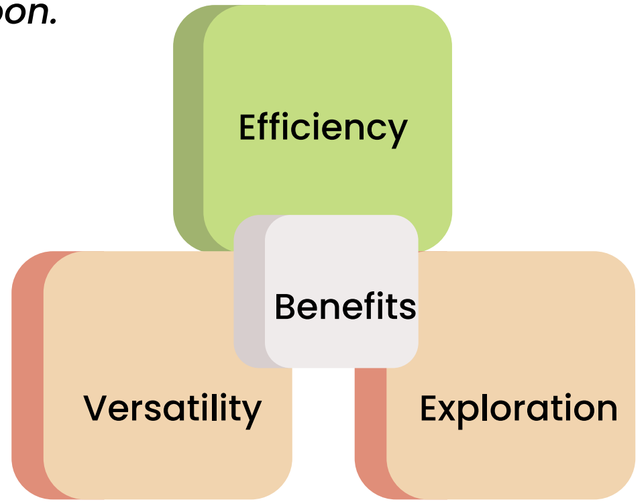
Zero-shot prompting is a technique used with large language models (LLMs) where the model can complete a task or generate an output without being explicitly trained on examples for that specific task.

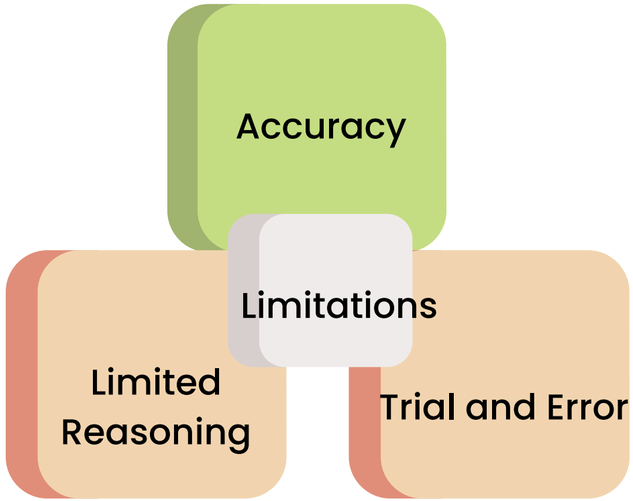
Crafting the Prompt: You provide the LLM with a prompt that instructs it on how to complete the task or what kind of output you desire. This prompt serves as a guide, conveying the task and desired format without needing pre-trained examples.

L**everaging LLM Capabilities:** The LLM uses its knowledge of language structures, patterns, and relationships between words to understand the prompt and generate the desired output.

Example: Imagine you want an LLM to write a haiku (a three-line Japanese poem with a specific syllable structure). Instead of training it on a dataset of haikus, you could simply provide a prompt like "Write a haiku about a cat basking in the sun" and the LLM, understanding the concept of a haiku and the prompt's instruction, might generate something like:

Soft fur warms in light,
Golden paws tucked in delight,
Purrs fill the afternoon.





ASHIMA MALIK

Few-Shot Prompting

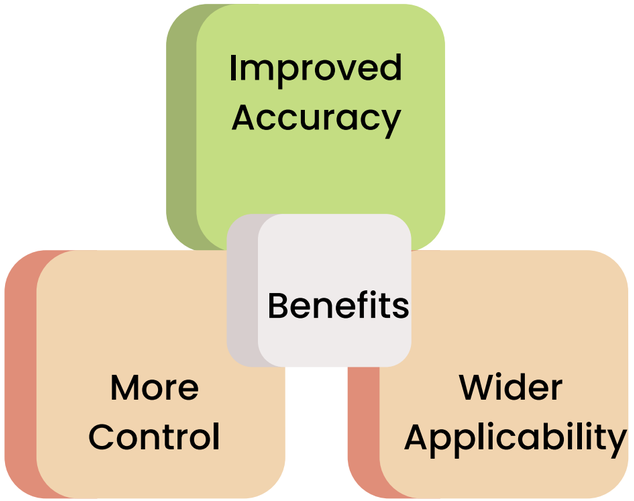
Crafting the Prompt: You create a prompt that instructs the LLM on the task and the desired format.

Providing Examples: In addition to the prompt, you present the LLM with 2-5 examples of the desired output format or task completion. These examples act as demonstrations for the LLM to learn from.

L**LM Learns from Examples:** The LLM uses the prompt and the few examples to grasp the task requirements and the style or format you're aiming for.

Generating Output: Based on the prompt, instructions, and the provided examples, the LLM generates its own response or completes the task.

Prompt: "Write a formal email requesting a meeting." Examples: (Provide 2-3 examples of formal email requesting a meeting).

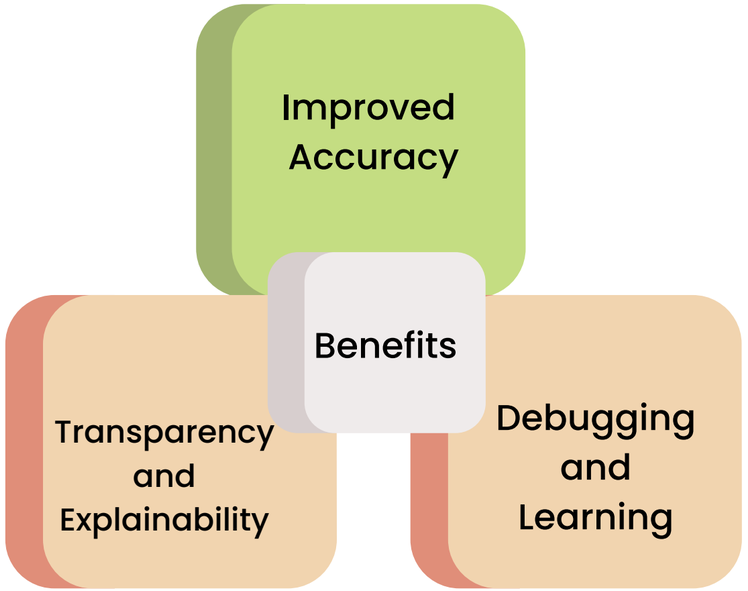


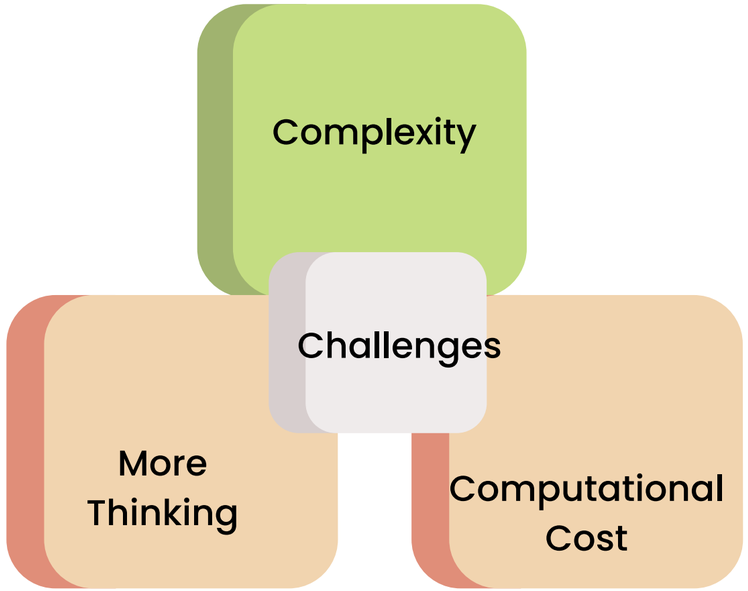
Chain-of-Thought Prompting

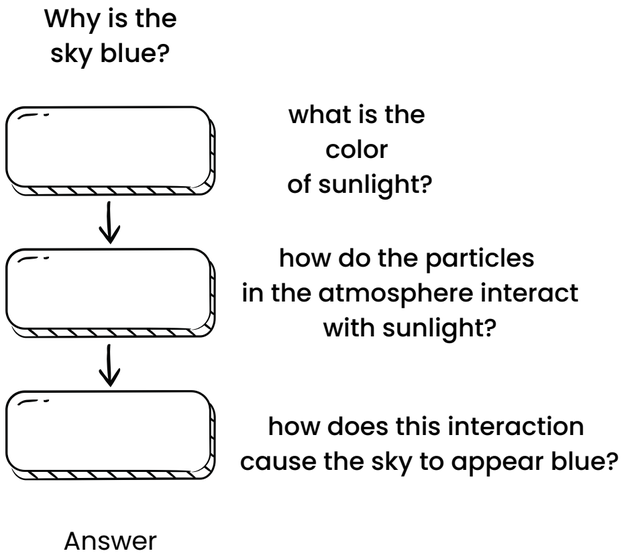
Chain-of-thought prompting (CoT) is a technique that encourages large language models (LLMs) to explain their reasoning process along with their final answer. This can be particularly useful when dealing with complex tasks or where understanding the LLM's thought process is crucial.

Prompt with Reasoning Steps: Unlike traditional prompts that simply ask a question or give an instruction, CoT prompts incorporate steps that guide the LLM to explain its thought process.

Example Prompt: Imagine you want an LLM to explain why the sky is blue. A traditional prompt might be: "Why is the sky blue?" A CoT prompt could be: "**Let's think step by step: Why is the sky blue? First, what is the color of sunlight? Next, how do the particles in the atmosphere interact with sunlight? Finally, how does this interaction cause the sky to appear blue?**"







ASHIMA MALIK

Self-consistency Prompting

It builds upon the idea of chain-of-thought prompting (CoT) but focuses on achieving a consensus among multiple outputs generated by the LLM itself.

Prompting the LLM: You provide the LLM with the prompt for the task or question you want answered.

Generating Multiple Outputs: Instead of relying on a single response, the LLM is prompted to generate the answer several times (typically 2-5 iterations).

Identifying Consistent Response: The outputs from each iteration are analyzed to identify the most common response across the attempts. This common response is considered the most self-consistent and reliable answer.

Prompt: Imagine you want to know the capital of France.

Iteration 1: LLM Response: "The capital of France is Paris."
Iteration 2: LLM Response: "Paris is the capital city of France."
Iteration 3: LLM Response: "France's capital is Paris."

Outcome: Since "Paris" emerged as the consistent answer across all iterations, it's considered the most reliable response through self-consistency prompting.

Benefits: Reduced Errors, Improved Confidence, Complements CoT

Limitations: Computational Cost, Not Guaranteed Consistency

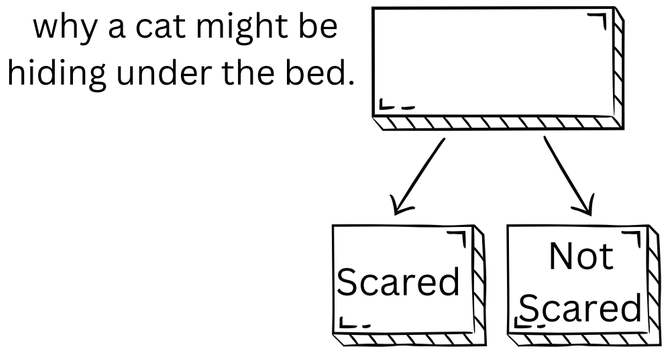
ASHIMA MALIK

Tree of Thoughts (ToT) Prompting

Similar to CoT, ToT aims to encourage LLMs to explain their reasoning process.
However, ToT takes a more sophisticated approach by creating a tree-like structure that allows the LLM to explore different reasoning paths and identify the most likely explanation.

Prompt with Exploration: The prompt guides the LLM to consider multiple possibilities and evaluate each one as a "thought" or node in the tree.
Branching Out: For each thought, the LLM can propose additional sub-thoughts or justifications, further branching out the tree.
Evaluation and Refinement: The LLM evaluates the plausibility of each thought and its sub-thoughts, potentially abandoning paths that seem illogical or irrelevant.
Final Answer and Reasoning: The LLM arrives at a final answer and provides a justification based on the most promising reasoning path in the tree.

Prompt: Let's explore why a cat might be hiding under the bed. First, consider if the cat feels scared. If scared, what might have caused it to feel that way? If not scared, are there other reasons a cat might hide under the bed?



Possible Reasoning Tree:

Scared (Yes):
Loud noise (e.g., thunder)
Vacuum cleaner
New person in the house
Scared (No):
Playing (hunting for imaginary prey)
Wants a quiet place to sleep

Final Answer: Based on the exploration, the LLM might conclude that the cat is scared due to a loud noise (or another reason under the "Scared" branch).

ASHIMA MALIK

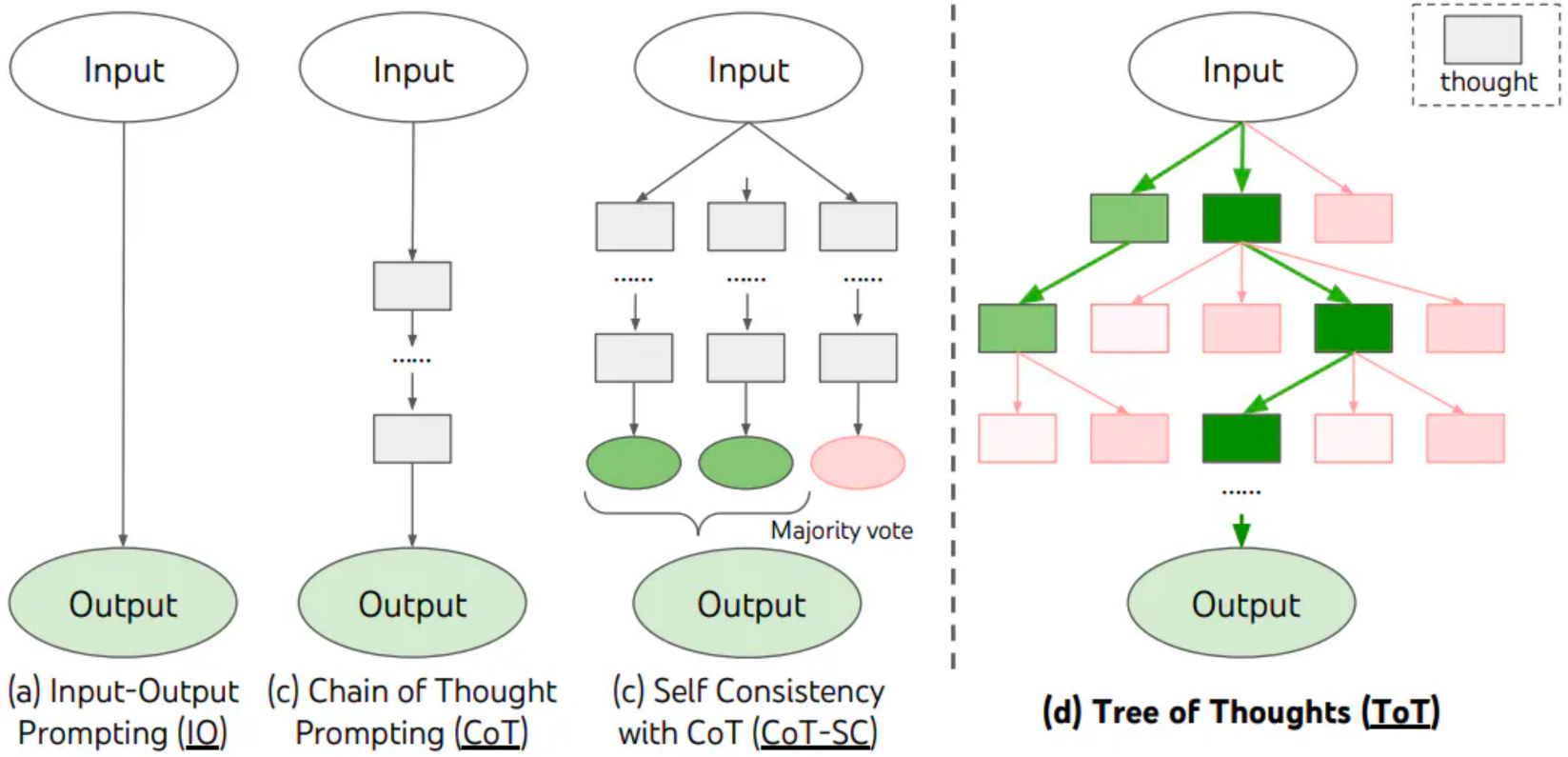
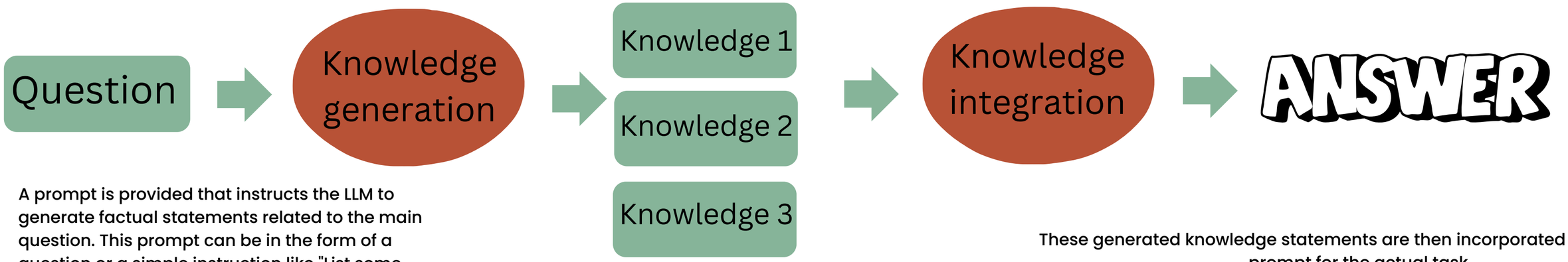


Image: https://www.promptingguide.ai/techniques/tot

Benefits: Improved Accuracy, Flexibility, Deeper Understanding

Challenges: Computational Cost, Complexity

Generated Knowledge Prompting



A prompt is provided that instructs the LLM to generate factual statements related to the main question. This prompt can be in the form of a question or a simple instruction like "List some relevant facts about...".

LLM generates a set of knowledge statements based on its understanding and the prompt's guidance

These generated knowledge statements are then incorporated into the main prompt for the actual task.
This provides the LLM with context and potentially improves the accuracy and relevance of its final output

Scenario: Imagine you want to ask an LLM a question about the historical impact of the printing press.

Traditional Prompt: "What was the historical impact of the printing press?"

GKP Prompt:

Generate Knowledge Statements: "List some ways information was shared before the printing press."
Main Task with Context: "Considering the knowledge statements you generated, explain the historical impact of the printing press."

Benefits: Improved Accuracy, Flexibility, Reduced Bias

Limitations: Accuracy of Generated Knowledge, Additional Processing Power

ASHIMA MALIK

Prompt chaining

Prompt chaining is a powerful technique used with large language models (LLMs) to achieve complex goals by breaking them down into smaller, more manageable tasks. It involves using the output from one prompt as the input for the next, guiding the LLM through a series of steps to reach the desired outcome.

- LLMs can struggle with overly complex prompts or instructions.
- Prompt chaining addresses this by providing a series of clear and concise prompts, one after another.
- Each prompt builds upon the previous one, gradually guiding the LLM towards the final goal.

How it Works:

-
- Break Down the Task: Analyze the desired outcome and identify the individual sub-tasks that lead to it.
- Craft Individual Prompts: Create clear and concise prompts for each sub-task, ensuring they logically connect and build upon each other.
- Chain the Prompts: Execute the prompts sequentially, feeding the output from one prompt as the input for the next.

Benefits:

- Improved Accuracy: Breaking down the task leads to more focused prompts, potentially improving the accuracy and relevance of the LLM's outputs.
- Efficiency: It can be more efficient than providing a single, overly complex prompt that the LLM might struggle with.
- Error Isolation: If an unexpected output occurs, it's easier to identify which step in the chain caused the issue.

Prompt chaining

Scenario: You want an LLM to write a creative story about a robot who falls in love with a human.

Without Prompt Chaining:

Prompt: "Write a story about a robot who falls in love with a human." (This prompt is quite broad and might lead to an ambiguous or unfocused story.)

With Prompt Chaining:

Prompt 1: "Describe a futuristic world where robots are commonplace and assist humans in their daily lives."
Prompt 2: "Create a character profile for a kind and curious robot who develops sentience and emotions."
Prompt 3: "Describe a human character, someone lonely yet apprehensive about robots."
Prompt 4: "Write a scene where the robot and the human meet and interact for the first time." (Based on the previous prompts)
Prompt 5: "As their encounters continue, show how the robot develops feelings for the human." (Building on the established scenario)
Prompt 6: "However, the robot faces challenges in expressing its love due to its limitations and societal norms." (Adding conflict based on previous prompts)
Prompt 7: "Write a concluding scene where the robot and the human find a way to connect, even if their love is unconventional." (Considering the challenges established earlier)

ReAct Prompting

ReAct Prompting is a technique for large language models (LLMs) that combines reasoning and interaction with external information sources. It goes beyond simply prompting the LLM for an answer and allows it to actively gather information and make decisions to complete a task.

ReAct prompts consist of three key elements: Thought, Action, and Observation.

- Thought: This represents the LLM's internal reasoning about the task or question at hand.
- Action: This instructs the LLM to take a specific action, such as searching for information or looking up a definition.
- Observation: This refers to the information or knowledge the LLM gains by performing the action.

How it Works:

- Initial Thought: The prompt starts with the LLM's initial understanding of the task or question.
- Action: Based on its initial thought, the LLM is instructed to take an action, such as "Search [keyword]" or "Lookup [concept]".
- Observation: The LLM interacts with an external information source (e.g., Wikipedia) and retrieves relevant information.
- Refined Thought (Optional): The LLM incorporates the retrieved information (Observation) into its reasoning and might formulate a more refined thought.
- Answer Generation: Based on the overall reasoning process (Thoughts, Actions, Observations), the LLM generates its final answer or completes the task.

ASHIMA MALIK

ReAct Prompting

Scenario: You want an LLM to answer the question "What is the capital of France?"

Traditional Prompt: "What is the capital of France?" (The LLM might rely solely on its internal knowledge, which might not always be accurate)

ReAct Prompt:

Thought 1: I need to find the capital of a country.
Action 1: Search [countries and their capitals].
Observation 1 (Retrieved Information): A table listing countries and their capitals, with France listed as "Paris".
Answer: The capital of France is Paris.

Benefits:

- Improved Accuracy: By actively seeking information, the LLM can access and utilize external knowledge to generate more accurate and well-informed outputs.
- Flexibility: ReAct prompting can be applied to various tasks that require information gathering, decision-making, and reasoning.
- Explainability: The reasoning steps (Thoughts) can provide insights into the LLM's thought process behind its final answer.

ReAct Prompting is a powerful technique for LLMs to go beyond their internal knowledge and leverage external information for more comprehensive and accurate results.

ASHIMA MALIK

Multimodal CoT Prompting

- Prompt with Text and Images: The prompt combines textual instructions with relevant images related to the task.
- Reasoning with Multimodal Cues: The LLM uses both the textual prompt and the visual information from the image(s) to understand the task and generate reasoning steps.
- Example Reasoning Steps: The reasoning steps might involve describing the image content, explaining the relationships between elements in the image, and how they connect to the textual prompt.
- Answer Generation: Based on the multimodal reasoning process, the LLM generates its answer or completes the task.

Introduces additional modalities like images or audio alongside textual prompts.
This allows the LLM to leverage richer information sources for its reasoning and potentially generate more accurate and insightful outputs.

Benefits:

- Improved Reasoning: By processing information from text and images together, the LLM can potentially develop a more comprehensive understanding of the task or situation.
- Reduced Ambiguity: Visual information can help clarify ambiguities in textual prompts, leading to more focused and relevant reasoning.
- Applicability to Broader Tasks: Multimodal CoT can be applied to tasks that naturally involve visual elements, like image segmentation or scene description.

Limitations:

- LLM Capabilities: The effectiveness of Multimodal CoT depends on the LLM's ability to process and reason with multimodal information.
- Data Availability: Training LLMs for effective multimodal reasoning requires large datasets that include both text and image data relevant to the task.

ASHIMA MALIK

Multimodal CoT Prompting

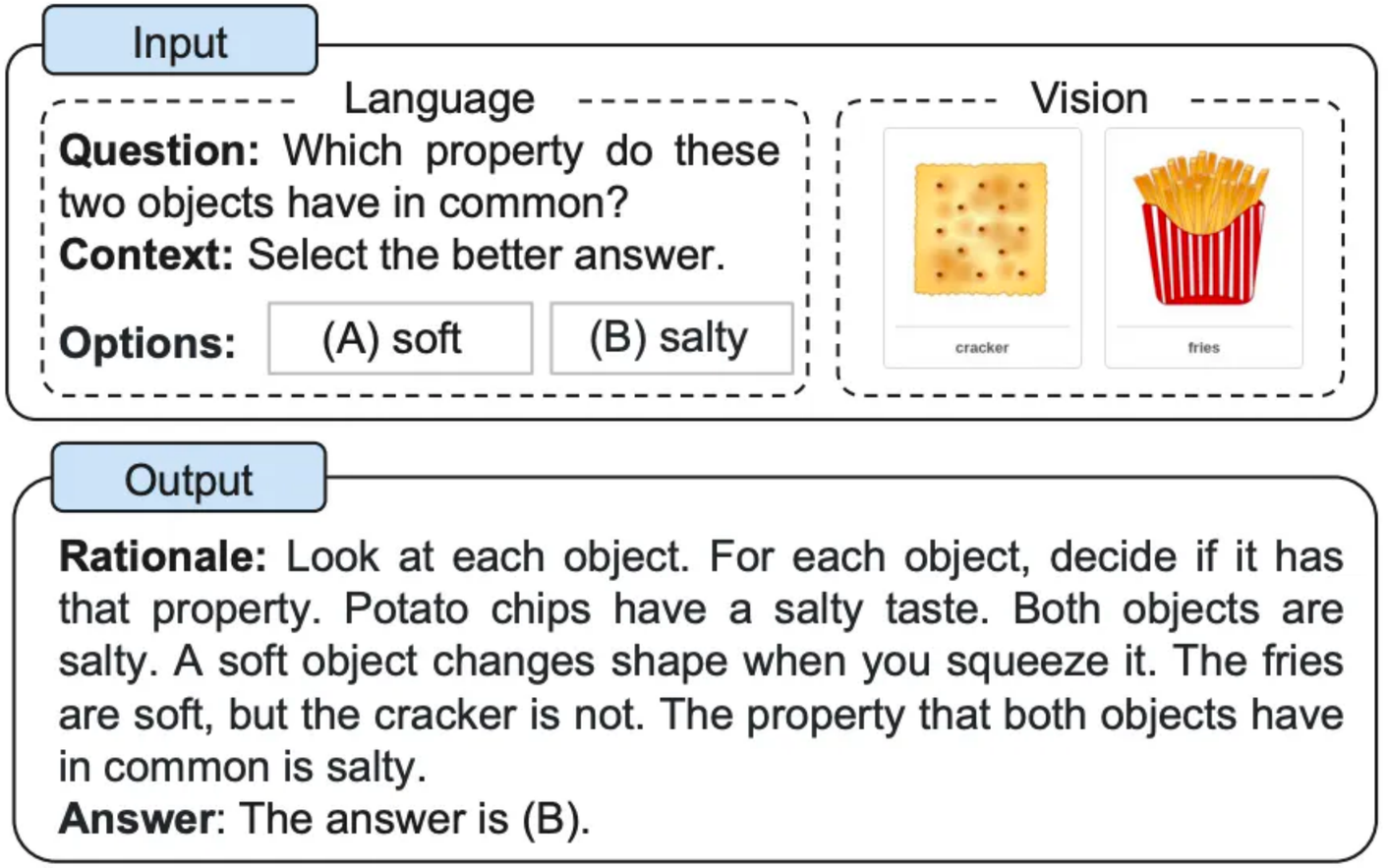


Figure 1. Example of the multimodal CoT task.

Image: https://www.promptingguide.ai/techniques/tot