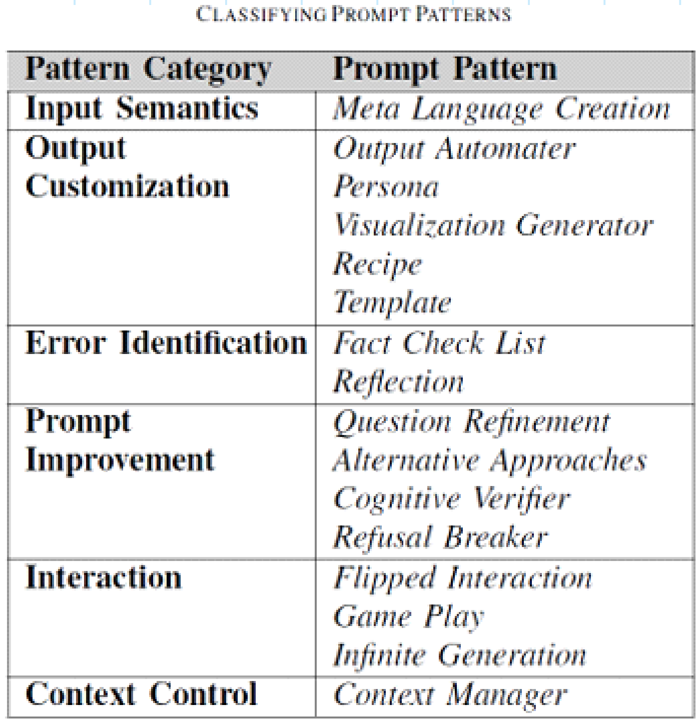
Prompt Engineering

17 December 2024 12:05

Prompt Engineering Patterns

Software Patterns	Prompt Patterns
Reusable solution to a recurring problem	Provide a codified approach to solving common software development challenges



CLASSIFYING PROMPT PATTERNS

Pattern Category	Prompt Pattern
Input Semantics	*Meta Language Creation*
Output Customization	*Output Automater* *Persona* *Visualization Generator* *Recipe* *Template*
Error Identification	*Fact Check List* *Reflection*
Prompt Improvement	*Question Refinement* *Alternative Approaches* *Cognitive Verifier* *Refusal Breaker*
Interaction	*Flipped Interaction* *Game Play* *Infinite Generation*
Context Control	*Context Manager*

Input Semantics:

How an LLM understand the input and translates that into something it can use to generate an output

1. **Meta language creation:**
    a. Create custom language for LLM to understand
    b. Useful when default language isn't good enough for conveying user's ideas
    c.

When I say X, I'm referring to/ I want you to do Y	When I say BOB, I'm talking about Spectral Decomposition

Output Customization:

Focuses on constraining or tailoring the properties of the output generated by the LLM

1. **Output Automater Pattern:**
    a. Allows the user to create scripts that can automate any tasks that output suggest user to perform
    b.

Produce an executable artefact of type X that will automate these steps	Usually a code

2. **Persona Pattern:**
    a. Give the LLM a role to play while generating an output
    b.

Act as X	From now on, act like my mom and give me the recipe for bread

3. **Visualization Generator Pattern:**
    a. Allows the user to generate visualizations by providing textual output that is then fed into other AI based image generators
    b.

Generate X so that I can provide it to tool	When I ask you to visualize something,

b.

Y to visualize it	return a DALL E prompt

4. **Recipe Pattern:**
    a. Allows the user to obtain a sequence of steps to realize the end result
5. **Template pattern:**
    a. Allows user to specify the template for the output

b.

Preserve the formatting and template that I provide	Generate a name and job title for Bob

Interaction:

1. **Flipped Interaction Pattern:**
    a. Get LLM to ask questions to obtain info it needs to perform tasks
    b. LLM drives conversation

c.

I would like you to ask me questions to achieve X	Ask me questions about my project. When you have enough information, give me the python script

2. **Infinite Generation Pattern:**
    a. Automatically generate a series of outputs without having to reenter a generator prompt each time

b.

I would like you to generate output forever, X outputs at a time	I want you to generate a name and job title until I say stop.

Prompt Improvement:

1. **Question Refinement Pattern:**
    a. Engage LLM in prompt engineering process
    b. Ensure that LLM always suggests a better or more refined version of original question

c.

Within scope X, suggest a better version of the question	From now on, when I ask a question, suggest a better version that incorporates information about Bob

2. **Alternative Approaches Pattern:**
    a. Ensure the LLM offers alternative ways to accomplish a task so that the user doesn't pursue approaches they're familiar with only

b.

Within scope X, if there's alternative way to accomplish the same thing, list the best alternate approaches	When I ask you to deploy a cloud application, if there are alternative services that do it better, list them

3. **Cognitive Verifier Pattern:**
    a. LLMs reason better when the question is subdivided into additional questions that provide answers combined into an overall answer to the original question
    b. Make LLM always subdivide questions

c.

When you are asked a question, generate a number of questions that help accurately answer the question	When I ask you about Bob, generate three more questions that will help you get a more accurate answer

Error Identification

1. **Fact Check List Pattern:**
    a. Ensure that LLM lists facts present in the output and form an important part of the statements

b.

Generate a set of facts that are contained in the output	From now on, when you generate an answer, create a set of facts that the answer depends on and fact-check it

2. **Reflection Pattern:**
    a. Ask model to explain the rationale behind the answers it gives
    b. Allows users to assess the model's validity

c.

Whenever you generate an answer, explain the reasoning and assumptions	When you provide an answer, list the reasons and assumptions behind your

behind your answer	selection of rationale

Context Control:

1. **Context Manager:**
   a. Enable users to specify or remove context for a conversation
   b.

Within scope X, consider Y and ignore Z	When analyzing this code, please consider only the security aspects

Working with Text

Text Modifier

1. **The Elaborator:**
   a. Adds relevant detail to sparse text
   b.

Yesterday was fun	Add more detail to text	Yesterday was the best day of my life, I am so happy I exist

2. **The Corrector:**
   a. Fixes grammatical errors in text
   b.

I is cool	Correct the grammar in the text	I am cool

3. **The Expander:**
   a. Makes text long by adding relevant details
   b.

Yesterday I baked cookies	Please expand this sentence with more relevant detail	Yesterday I baked two dozen chewy chocolate chip cookies using my grandmother's secret recipe. I added extra chocolate chips for extra flavor and enjoyment.

4. **The Paraphraser:**
   a. Rephrases text with different vocabulary and syntax
   b.

The massive ursine mammal	Please rephrase	The huge bear

Text Generator

1. **The Generator:**
   a. Prompt introduces new text from scratch on a given topic
   b.

Dogs	Please write a story about dogs	The little puppy Waggles was so excited to go to the park…

2. **The Completer:**
   a. Fills in sections of incomplete text
   b.

I am a	Fill in the blanks	I am a cool girl

Text Analyzer

1. **The Classifier**:
   a. Categorizes input into predefined classes
   b.

The movie was nice	Classify as bad or good	Good

2. **The Extractor:**
   a. Extracts key pieces of info
   b.

My name is bob and I'm 25	Extract the name and age	Bob, 25

3. **The Fact Checker:**
   a. Validates accuracy of facts in text
   b.

The first PM was Gandhi	Check text for factual accuracy	Wrong. The first PM was Nehru

4. **The Summarizer:**

a. Distills key information

b.

Artificial intelligence has seen rapid advances in recent years. New techniques like deep learning have enabled AI systems to achieve human-level performance on tasks like image recognition, speech processing and language translation...	Summarize in one line	Recent advances in deep learning have improved AI systems' abilities in areas like computer vision and natural language processing.

Text Restructurer

1. **The Reformatter:**
   a. Restructures the input into a different format

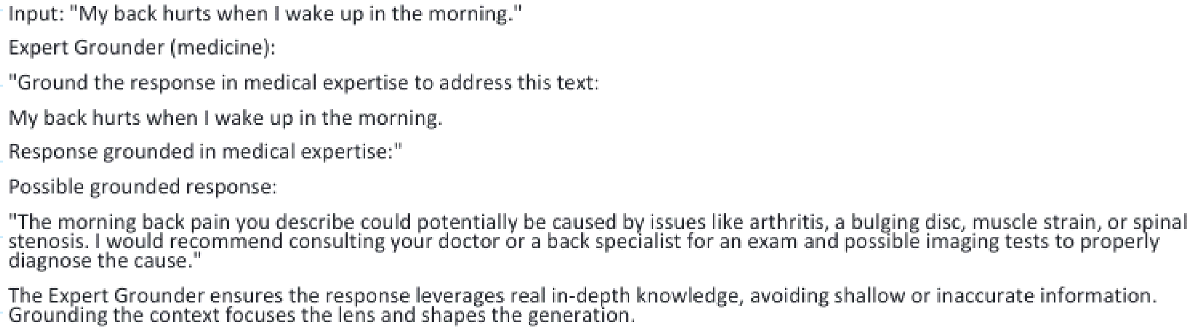
   b.

John, 25, New York	Reformat into a sentence	John is a 25-year-old who lives in New York.

Knowledge Grounder

1. **The Expert Grounder:**
   a. Leverage knowledge from a specific domain to shape the response

   b.



Input: "My back hurts when I wake up in the morning."

Expert Grounder (medicine):

"Ground the response in medical expertise to address this text:

My back hurts when I wake up in the morning.

Response grounded in medical expertise:"

Possible grounded response:

"The morning back pain you describe could potentially be caused by issues like arthritis, a bulging disc, muscle strain, or spinal stenosis. I would recommend consulting your doctor or a back specialist for an exam and possible imaging tests to properly diagnose the cause."

The Expert Grounder ensures the response leverages real in-depth knowledge, avoiding shallow or inaccurate information. Grounding the context focuses the lens and shapes the generation.

2. **The Context Binder:**
   a. Connects separate pieces of info together

   b.

I am a girl. I am cool. I am Namita.	Create a cohesive sentence	I am Namita, a cool girl.

Other cool patterns:

1. **The Balancer-** add counter arguments
2. **The Disambiguator-** resolve ambiguous words
3. **The Metaphorizer-** use metaphors to illustrate abstract ideas
4. **The Ironizer-** Add irony or sarcasm
5. **The Euphemizer-** Replace negative words with gentler terms
6. **The Clickbaiter-** Adds exaggerated and questionable claims
7. **The Roaster-** Mocking or sarcastic language
8. **The Flatterer-** Adds complimentary language

Prompt Engineering Techniques

The practice of crafting effective prompts to get the most out of LLMs.

Structured Input and Output:

- Helps model better understand the task and input data
- Makes model's responses easier to programmatically parse and integrate downstream
- When:
  - To simplify model outputs into databases and APIs
  - Processing data like product info, reviews

Prefilling Model Response:

- Guarantees that the response will start with the provided text
- Useful for prompting a specific output format and simplifying parsing
- When:

    - Enforce consistent output format
    - Output needs to be parsed or deserialized in a certain way

N-shot Prompting

- Provide multiple input-output examples and demonstrates to the model how to perform the task and the required output format
- When:
    - Improve output quality and consistency
    - Specific format or style needs to be followed

Chain of Thought Prompting

- Encourages model to break down reasoning into intermediate steps before providing final output
- When:
    - Tasks with multiple steps, reasoning, calculations
    - Improve transparency and interpretability
    - Output factuality and faithfulness of input is critical

Temperature

- Controls creativity of outputs
- Low temp = focused, deterministic response
- High temp = diverse, stochastic
- When:
    - Set for use case
    - Dial up and down based off focus v creativity levels

Mitigating Hallucination

- Hallucination = fabricate something not present in input
- Reduce by asking model to express uncertainty when required
- When:
    - Questions answering
    - Knowledge retrieval tasks
    - Factual accuracy is critical

Prompt Chaining

- Output of one prompt is used as input to subsequent prompt
- Allows guiding model through multistep generative process to iteratively refine and improve output
- When:
    - Complex generative tasks that can be broken into steps
    - Iteratively refine and improve output
    - Single prompt is too complicated

Prompt Augmentation

- Inject relevant into or context into prompt itself
- When:
    - Content required highly specific or niche knowledge
    - Customize model output
    - Default knowledge is outdated

Prompt Ensembling

- Combine outputs from multiple prompts to produce a final response that incorporates the best aspects of each individual's prompt
- Why?
    - Leverage strengths and mitigate weakness

Prompt Based Finetuning

- Integrate prompts into model finetuning process
- Format training data in same question-answer format as target prompts
- Model follows specific instructions and reproduces prompted behavior reliably
- When:
  - Large enough dataset
  - Reliable follow specific interaction pattern
  - Bake in certain prompts into LLM behavior
  - Faster inference

Prompt Tuning

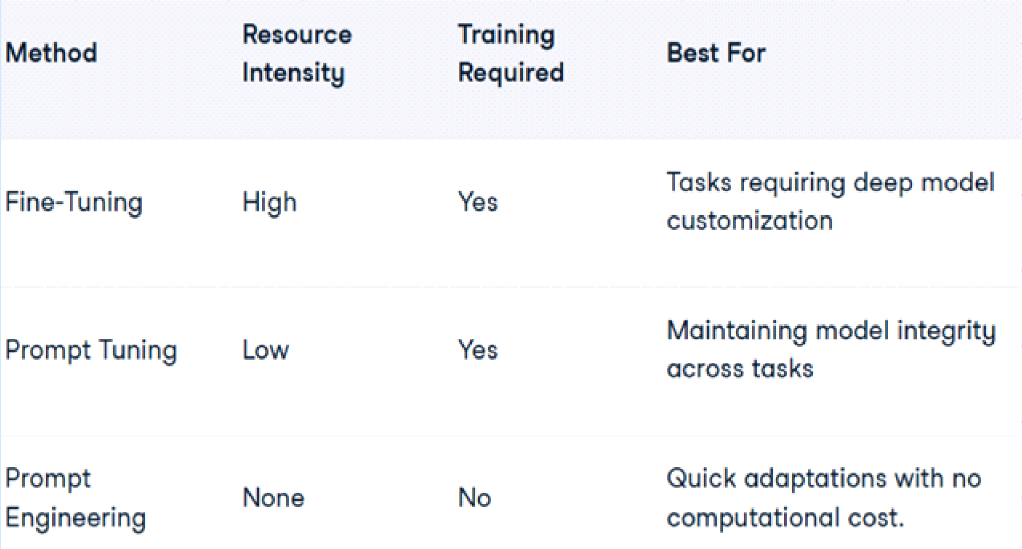
- Technique to enhance performance of pre-trained model without altering the core architecture
- Modify deep structural weights of the model
- Adjust prompts and guide model's response
- Based on intro of soft prompts (set of tunable params inserted at the beginning of input sequence)

Finetuning

- Most resource intensive
- Retrain model on specific dataset for a specific purpose
- Adjusts weights of pretrained model and optimizes for detailed nuances of data
- Requires substantial comp resources
- High risk of overfitting

Prompt Engineering

- No training or retraining
- Based entirely on user designed prompts for the model
- Requires nuanced understanding of the model's capabilities and leverages intrinsic knowledge available in model



Method	Resource Intensity	Training Required	Best For
Fine-Tuning	High	Yes	Tasks requiring deep model customization
Prompt Tuning	Low	Yes	Maintaining model integrity across tasks
Prompt Engineering	None	No	Quick adaptations with no computational cost.

Advanced Framework for LLM Applications

LangChain

- Open source
- Simplify development of applications using LLMs
- Can build LLM powered apps that are:
  - Data aware (can connect with external data bases)
  - Modular (common interfaces around prompts, models, all that)

    - Customizable (supports different LLMs)
    - Scalable (easy to deploy on various platforms and apps can scale up or down as needed)
- Use cases:
    - Chatbots
    - QA systems
    - Summarization

LlamaIndex

- Ingest and structure private or domain specific data to be used with LLMs
- It allows:
    - Connecting to data sources
    - Indexing the data for retrieval and association
    - Retrieving relevant data for LLM prompts
    - Caching indexed data for low latency access
- Client libraries: easy to integrate into apps with LangChain

	LangChain	**LlamaIndex**
Primary Focus	NLP applications	Search and retrieval
Strength	Modular tools, integrations	Data indexing, querying
Use Cases	Chatbots, customer support	Document retrieval
Recommendation	Complex workflows	Quick data access