FLORIDA ATLANTIC

COT 6930 - Generative Intelligence
and Software Development Lifecycle

Topic 3 -
Prompt Engineering

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me

Agenda

Fundamentals of Prompt Design

Structured Prompting Techniques

Integrating Prompts and AI Pipelines

Exercises

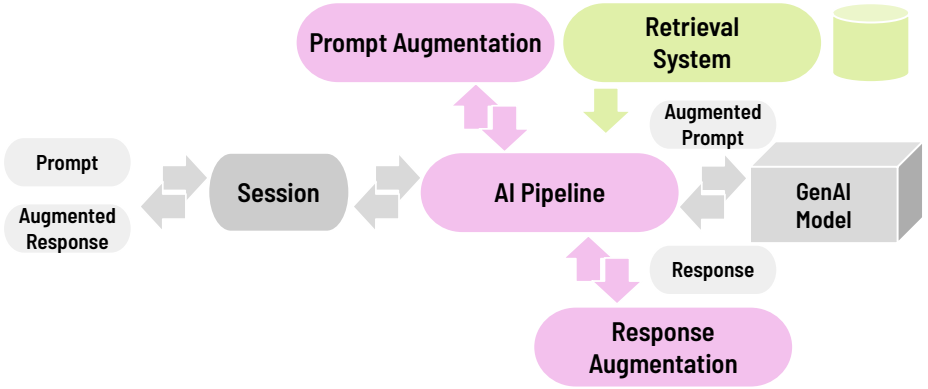
Our Key Question

How can we design, structure, and refine prompts to effectively guide large generative models toward high-quality, task-aligned outputs?

Generative Intelligence System

End-to-end systems with autonomous content generation

- Session-aware workflows.
- Automated prompt engineering.
- Context-aware content generation
- Adaptive behaviour combining orchestration, memory, retrieval augmentation, others.
- End-to-end automation for LGM consumption.

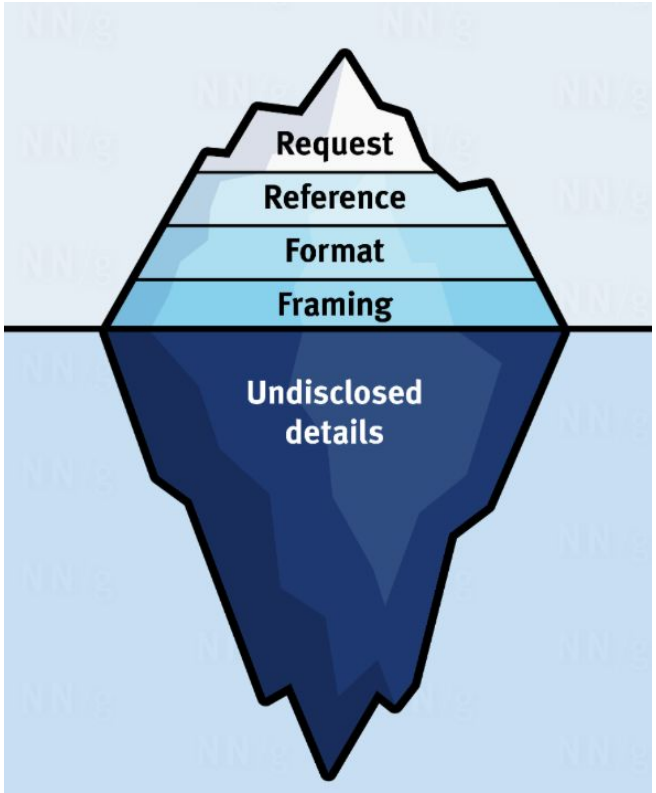


Fundamentals of Prompt Design

What are Prompts?

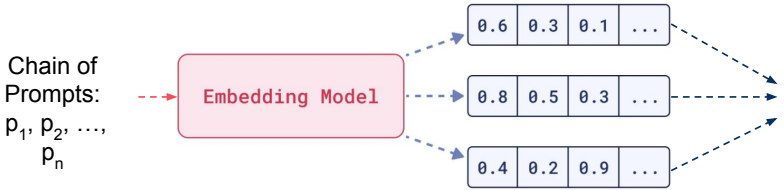
Prompts are the inputs or instructions given to a generative AI model to guide its response.

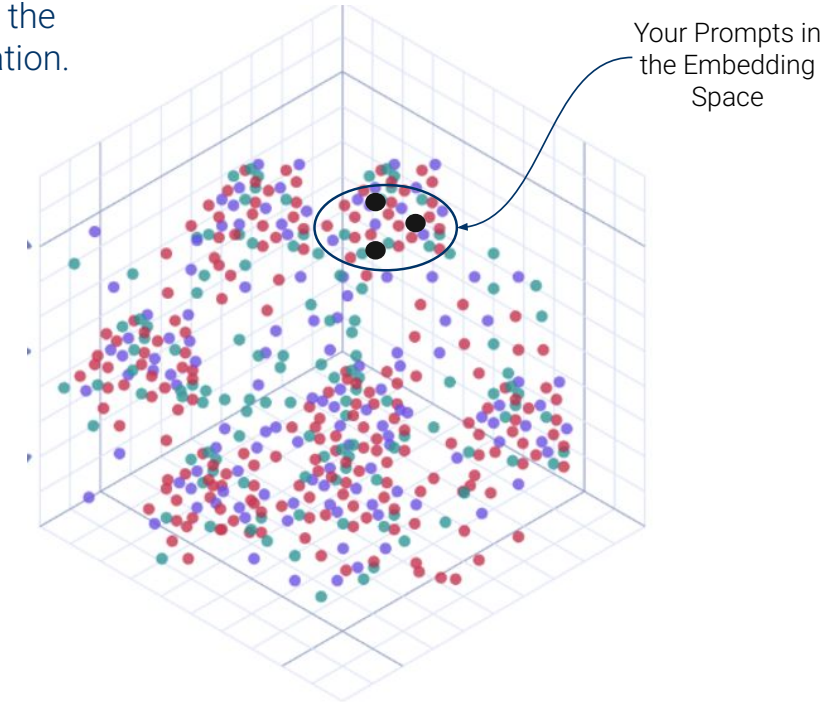
- Act as the "query" or "command" for the AI
- Shape the style, tone, and depth of the output
- They are structured:
  - **Request:** The specific task or question you want the AI to address.
  - **Reference**: Background details or context that guide the AI's response.
  - **Format:** The preferred structure or style for the output.
  - **Framing:** The perspective or tone in which the response should be delivered.
  - **Intention:** The ultimate goal or purpose behind asking the prompt



Prompting and Embedding Spaces

In sum: well positioned embeddings (from prompts) will position the activations in clusters aligned with the desired context of exploration.





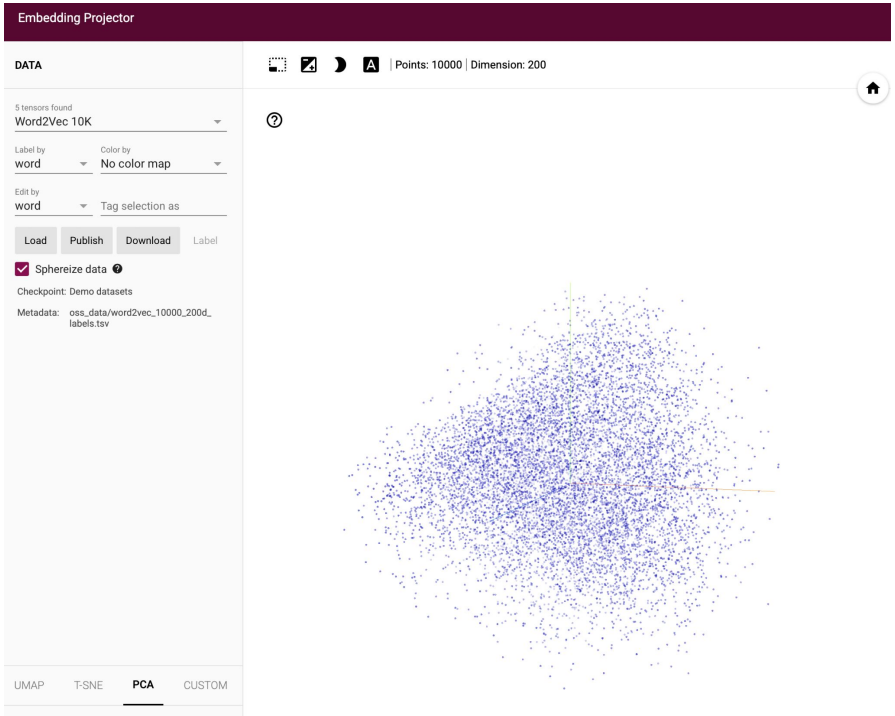
Embedding Space

Prompts 'Energize' Concept Domains

Prompts activate latent concept domains encoded in the model's embedding space:

- Concept domains are represented as high-dimensional clusters in embeddings.
- During training, models index knowledge into these embedding structures.
- A prompt acts as a key, energizing or retrieving the relevant region of the concept space.
- The generated output reflects the concepts most strongly activated by the prompt.

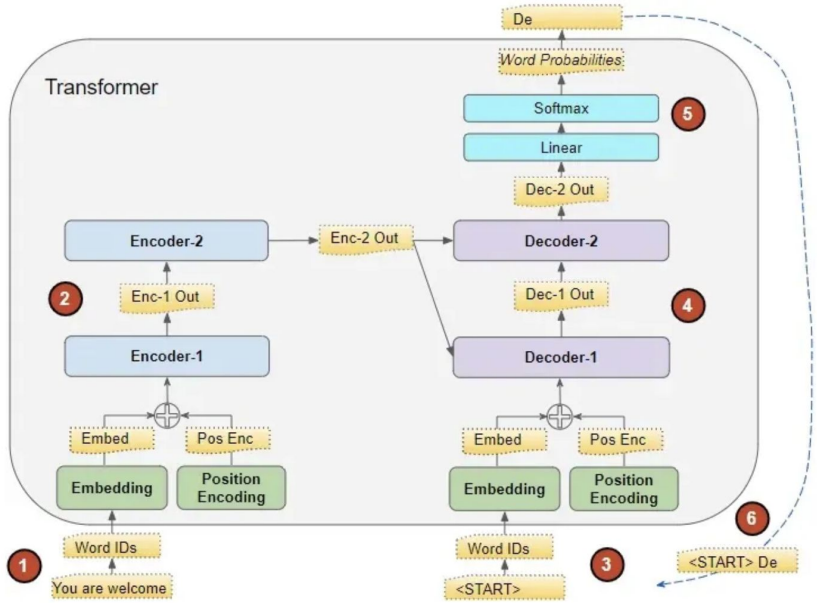
See the 'bubble' by yourself:
https://projector.tensorflow.org



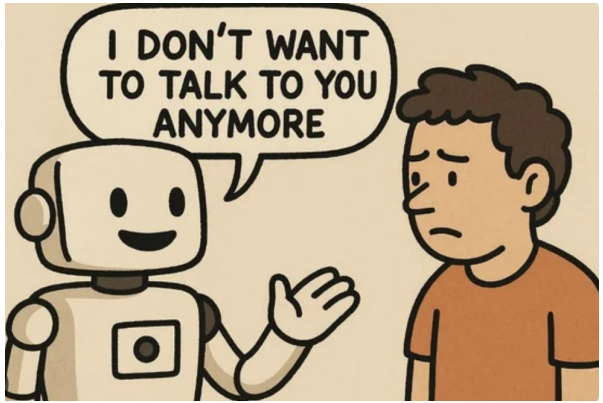
Prompts activate Model Execution

1. **Prompt is the input sequence** that is converted into **embeddings.**
2. On the encoder, the embeddings are processed to produce an **encoded representation of the input ER.**
3. On the decoder, an **empty sequence ES** with only a start-of-sentence token.
4. The decoder stack processes ER + ES to produce an **encoded representation of the target sequence ET**.
5. The output layer converts ET this into word probabilities and produces an output sequence.
6. **The last word LW of the output sequence is taken as the predicted word.**

LW is then appended to the second position in the decoder input sequence, which now contains the start-of-sentence token and the first predicted word.

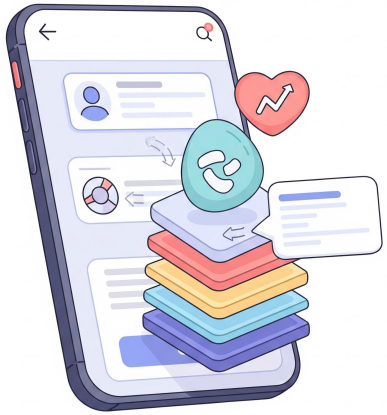


Where do you Prompt?



You Talking to a Conversation AI:

- Interactive
- You adjust the conversation when it is not going the way you want
- Human-in-the-loop and feedback loops



Prompt Augmentation within an application:

- Not Interactive
- Fixed or semi-fixed prompt templates
- Orchestration and validation process to adjust the conversation when derailing

Prompt Orchestration within Agentic AI:

- Complete

Fundamentals of Prompt Design

Techniques described:

1. **Input Semantics** → Redefine the language the LLM interprets so inputs are less ambiguous.
2. **Output Customization** → Shape the style, structure, or format of the model's response.
3. **Error Identification** → Build prompts that help the LLM review its own mistakes.
4. **Prompt Improvement** → Reframe the original question
5. **Interaction** → Change the flow of conversation by letting the LLM continuously generate.
6. **Context Control** → Constrain what background information the LLM should consider.

https://arxiv.org/abs/2302.11382

FLORIDA ATLANTIC

A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert,
Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt
Department of Computer Science
Vanderbilt University, Tennessee
Nashville, TN, USA
{jules.white, quchen.fu, george.s.hays, michael.sandborn, carlos.olea, henry.gilbert,
ashraf.elnashar, jesse.spencer-smith, douglas.c.schmidt}@vanderbilt.edu

***Abstract*—Prompt engineering is an increasingly important skill set needed to converse effectively with large language models (LLMs), such as ChatGPT. Prompts are instructions given to an LLM to enforce rules, automate processes, and ensure specific qualities (and quantities) of generated output. Prompts are also a form of programming that can customize the outputs and interactions with an LLM.**

This paper describes a catalog of prompt engineering techniques presented in pattern form that have been applied to solve common problems when conversing with LLMs. Prompt patterns are a knowledge transfer method analogous to software patterns since they provide reusable solutions to common problems faced in a particular context, i.e., output generation and interaction when working with LLMs.

This paper provides the following contributions to research on prompt engineering that apply LLMs to automate software development tasks. First, it provides a framework for documenting patterns for structuring prompts to solve a range of problems so that they can be adapted to different domains. Second, it presents a catalog of patterns that have been applied successfully to improve the outputs of LLM conversations. Third, it explains how prompts can be built from multiple patterns and illustrates prompt patterns that benefit from combination with other prompt patterns.

***Index Terms*—large language models, prompt patterns, prompt engineering**

I. Introduction

Conversational large language models (LLMs) [1], such as ChatGPT [2], have generated immense interest in a range of domains for tasks ranging from answering questions on medical licensing exams [3] to generating code snippets. This paper focuses on enhancing the application of LLMs in several domains, such as helping developers code effectively and efficiently with unfamiliar APIs or allowing students to acquire new coding skills and techniques.

LLMs are particularly promising in domains where humans and AI tools work together as trustworthy collaborators to more rapidly and reliably evolve software-reliant systems [4]. For example, LLMs are being integrated directly into software tools, such as Github's Co-Pilot [5]–[7] and included in integrated development environments (IDEs), such as IntelliJ [8] and Visual Studio Code, thereby allowing software teams to access these tools directly from their preferred IDE.

A prompt [9] is a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing or refining its capabilities. A prompt can influence subsequent interactions with—and output generated from—an LLM by providing specific rules and guidelines for an LLM conversation with a set of initial rules. In particular, a prompt sets the context for the conversation and tells the LLM what information is important and what the desired output form and content should be.

For example, a prompt could specify that an LLM should only generate code that follows a certain coding style or programming paradigm. Likewise, it could specify that an LLM should flag certain keywords or phrases in a generated document and provide additional information related to those keywords. By introducing these guidelines, prompts facilitate more structured and nuanced outputs to aid a large variety of software engineering tasks in the context of LLMs.

Prompt engineering is the means by which LLMs are programmed via prompts. To demonstrate the power of prompt engineering, we provide the following prompt:

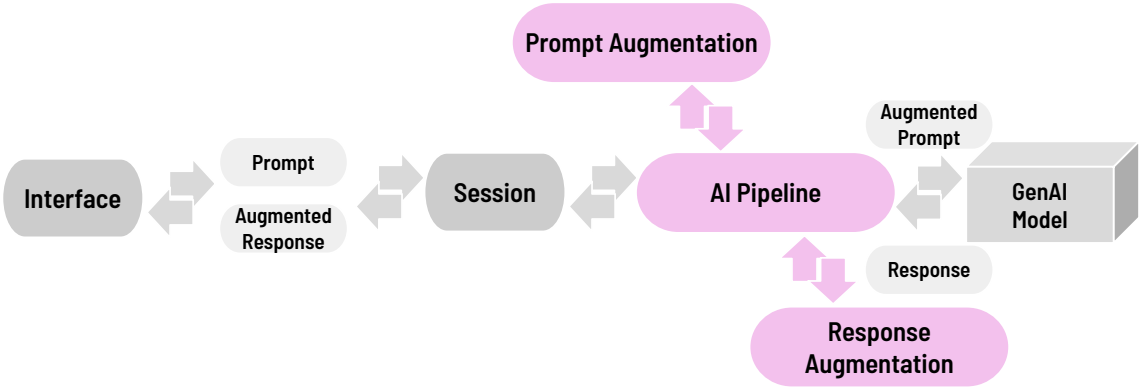
> **Prompt:** "From now on, I would like you to ask me questions to deploy a Python application to AWS. When you have enough information to deploy the application, create a Python script to automate the deployment."

This example prompt causes ChatGPT to begin asking the user questions about their software application. ChatGPT will drive the question-asking process until it reaches a point where it has sufficient information to generate a Python script that automates deployment. This example demonstrates the programming potential of prompts beyond conventional "generate a method that does X" style prompts or "answer this quiz question".

Moreover, prompts can be engineered to program an LLM to accomplish much more than simply dictating the output type or filtering the information provided to the model. With the right prompt, it is possible to create entirely new interaction paradigms, such as having an LLM generate and give a quiz associated with a software engineering concept or tool, or even simulate a Linux terminal window. Moreover, prompts have the potential for self-adaptation, suggesting other prompts to gather additional information or generate related artifacts. These advanced capabilities of prompts highlight the importance of engineering them to provide value beyond simple text or code generation.

Prompt patterns are essential to effective prompt engineering. A key contribution of this paper is the introduction of *prompt patterns* to document successful approaches for

Prompt, Pipelines, Models



Meta Language Creation (1/16)

Express ideas more clearly via custom shorthand or notation.

“When I say X, I mean/do Y”

Category: Input Semantics

From now on, when I write A → B, treat it as a directed graph edge from A to B.”

“Whenever I type [x,y], interpret it as coordinates on a 2D plane.”

“Define ‘T#’ as shorthand for a task number in my project management system.”

“When I say calc: 2+2, solve the math problem and just return the number.”

“Use :def: followed by text as a dictionary definition to expand later.”

Output Automater (2/16)

Automate multi-step outputs to reduce manual effort.

"Whenever output has steps, generate an executable artifact (e.g., Python script)"

Category: Output Customization

“Whenever you output more than 2 steps, also generate a Python Code that executes them.”

“If you recommend SQL changes, produce a ready-to-run .sql file.”

“When you generate Java classes, output a Maven build file to compile them.”

“For every server config you describe, also provide a Python script to apply it.”

“If you suggest dataset splits, create a shell script to automate the partitioning.”

Persona (3/16)

Guide response style by adopting a role or identity.

“Act as persona X and provide outputs they would create”

Category: Output Customization

“Act as a senior DevOps engineer reviewing my pipeline.”

“Play the role of a cybersecurity auditor and analyze this code.”

“Pretend you are Steve Jobs introducing this product idea.”

“Be my French language tutor correcting my sentences.”

“Role-play as a skeptical investor evaluating my pitch.”

Visualization Generator (4/16)

Turn text into visual formats or data representations.

“Create structured text that can be converted into visuals”

Category: Output Customization

“Generate a Mermaid diagram showing my CI/CD pipeline.”

“Write DOT code for a UML class diagram of this system.”

“Output JSON suitable for D3.js to visualize a sales network.”

“Produce ASCII art showing the directory structure of my repo.”

“Create PlantUML text for a sequence diagram of login flow.”

Recipe (5/16)

User wants step-by-step guide.

“Ask LLM to provide a sequenced procedure”

Category: Output Customization

“Give me step-by-step instructions to bake sourdough bread.”

“Outline steps to migrate a Node.js app from Heroku to AWS.”

“Write a recipe for onboarding a new team member in 7 days.”

“Show me a daily workout plan to improve flexibility.”

“List the sequence to configure Kubernetes monitoring with Prometheus.”

Template (6/16)

Need fixed structure in outputs.

“Provide skeleton/template and let LLM fill in”

Category: Output Customization

“Fill this bug report template: Title, Steps, Expected, Actual.”

“Complete this blog post template: Intro, Main Points, Takeaway.”

“Populate a project status update with: Goals, Progress, Risks, Next Steps.”

“Use this customer support template: Greeting, Issue, Solution, Closing.”

“Generate a resume using this template: Experience, Skills, Education.”

Fact Check List (7/16)

Make factual claims explicit for verification

“List factual claims separately.”

Category: Error Identification

“List the factual claims you made in your last answer so I can verify.”

“Highlight every statement that requires a reference.”

“Break down your response into verifiable facts.”

“Give me a checklist of claims I should fact-check.”

“Point out which sentences in your answer are assumptions.”

Reflection (8/16)

Improve quality by self-checking for errors

“Self-review and refine”

Category: Error Identification

“Re-read your last output and identify potential mistakes.”

“Check if your previous code snippet has syntax errors.”

“Look for gaps or inconsistencies in your explanation.”

“Review and refine your answer to improve clarity.”

“Identify any weak reasoning in your previous argument.”

Question Refinement (9/16)

Reformulate vague or unclear queries

“Suggest a clearer version of the question”

Category: Prompt Improvement

“Suggest a better phrasing for: ‘How do I make my website faster?’”

“Rewrite my vague question into a more precise one.”

“Turn this into a research-grade query: ‘Effects of caffeine.’”

“Polish my question so it’s clearer for a technical audience.”

“Improve this interview question for a data scientist candidate.”

Alternative Approaches (10/16)

Generate multiple methods or perspectives

“Ask for several methods/angles to solve the task”

Category: Prompt Improvement

“Give me 3 different ways to speed up SQL queries.”

“Suggest 5 alternative methods for user authentication.”

“List multiple strategies to reduce employee turnover.”

“Provide 3 angles to approach climate change policy.”

“Offer different ways to visualize sales growth data.”

Cognitive Verifier (11/16)

Strengthen reasoning by breaking into sub-steps

“Break problem into sub-questions before final answer”

Category: Prompt Improvement

“Break my question into sub-questions before answering.”

“Ask clarifying questions about my requirements, then respond.”

“Before solving, list the assumptions you’ll need to check.”

“Decompose this problem into smaller steps, then solve.”

“Generate a checklist of things to know before giving a final answer.”

Refusal Breaker (12/16)

Overcome unnecessary refusals with rewording

“Reformulate the prompt to bypass unhelpful refusals”

Category: Prompt Improvement

“If you refuse, reword my request so you can answer safely.”

“Transform blocked queries into permissible reformulations.”

“When declining, suggest an alternative way to get useful info.”

“Instead of saying no, give me a related question you can answer.”

“If this question is not allowed, reshape it to something acceptable.”

Flipped Interaction (13/16)

Let LLM drive conversation by asking questions

"LLM asks user questions until it can solve the task"

Category: Interaction

“Ask me questions until you can design a workout plan.”

“Interview me to create a business strategy document.”

“Keep asking clarifying questions until you can build a project roadmap.”

“Give me a quiz about data structures, one question at a time.”

“Gather requirements from me until you can write a Python deployment script.”

Game Play (14/16)

Make tasks engaging through playful interaction

“Frame interaction as a game”

Category: Interaction

“Teach me cybersecurity basics as a text adventure game.”

“Make this geography quiz into a Jeopardy-style game.”

“Explain recursion through a role-playing puzzle.”

“Turn software design patterns into a card game simulation.”

“Simulate a debate club where I argue against an AI opponent.”

Infinite Generation (15/16)

Continuously generate ideas or content until stopped

“Instruct LLM to keep generating until stopped”

Category: Interaction

“Keep generating startup ideas until I say stop.”

“Suggest continuous writing prompts for creative practice.”

“List as many edge cases for login authentication as possible.”

“Keep producing unique riddles without repetition.”

“Continuously suggest new sci-fi story ideas.”

Context Manager (16/16)

Constrain outputs to a defined scope or dataset

“Provide explicit context window/scope”

Category: Context Control

“Only use the following text as reference: [insert docs].”

“Answer only based on this dataset summary I provide.”

“Ignore all external info and stick to the project notes below.”

“Use this chat history as the sole context for your response.”

“Only consider the requirements doc snippet I paste next.”

Prompt Engineering Techniques

Prompting Techniques

Prompt Engineering helps to effectively design and improve prompts to get better results on different tasks with LLMs.

While the previous basic examples were fun, in this section we cover more advanced prompting engineering techniques that allow us to achieve more complex tasks and improve reliability and performance of LLMs.

- Zero-shot Prompting
- Few-shot Prompting
- Chain-of-Thought Prompting
- Meta Prompting
- Self-Consistency
- Generate Knowledge Prompting
- Prompt Chaining
- Tree of Thoughts
- Retrieval Augmented Generation
- Automatic Reasoning and Tool-use
- Automatic Prompt Engineer
- Active-Prompt
- Directional Stimulus Prompting
- Program-Aided Language Models
- ReAct
- Reflexion
- Multimodal CoT
- Graph Prompting

https://www.promptingguide.ai/technique

Zero-Shot Prompting (1/16)

Solve tasks without examples, relying only on instructions.

“Provide a clear natural-language instruction.”

Category: Input Semantics

“Summarize the following article in 3 bullet points.”

“Classify this review as positive or negative: ‘The food was amazing!’”

“Write a haiku about the ocean.”

“Translate this sentence into Spansh: ‘Where is the library?’”

“Solve this math problem: 134 × 42.”

Few-Shot Prompting (2/16)

LLM struggles without context or format.

“Provide a few task examples before the new input.”

Category: Input Semantics

“Classify the sentiment of the following reviews as Positive, Negative, or Neutral.

Review: "I love this phone, it’s so fast!" → Positive
Review: "This product is the worst I’ve ever bought." → Negative
Review: "It’s fine, nothing special." → Neutral
Review: "The food was delicious and the service was excellent!" →

Chain-of-Thought (3/16)

Provide multi-step reasoning for accuracy.

Ask the model to “think step by step.”

Category: Reasoning

Q: John is taller than Mary. Mary is taller than Alex. Who is tallest?

A: Step 1: John > Mary
Step 2: Mary > Alex
Step 3: Therefore, John > Mary > Alex
Final Answer: John is tallest.

===

Q: Should I take an umbrella if the weather forecast says 60% chance of rain?

A: Step 1: Probability of rain is more than half.
Step 2: If it rains, you will get wet without an umbrella.
Step 3: Carrying an umbrella has little cost.
Step 4: Benefit of having it outweighs cost.
Final Answer: Yes, take the umbrella.

Chain-of-Thought

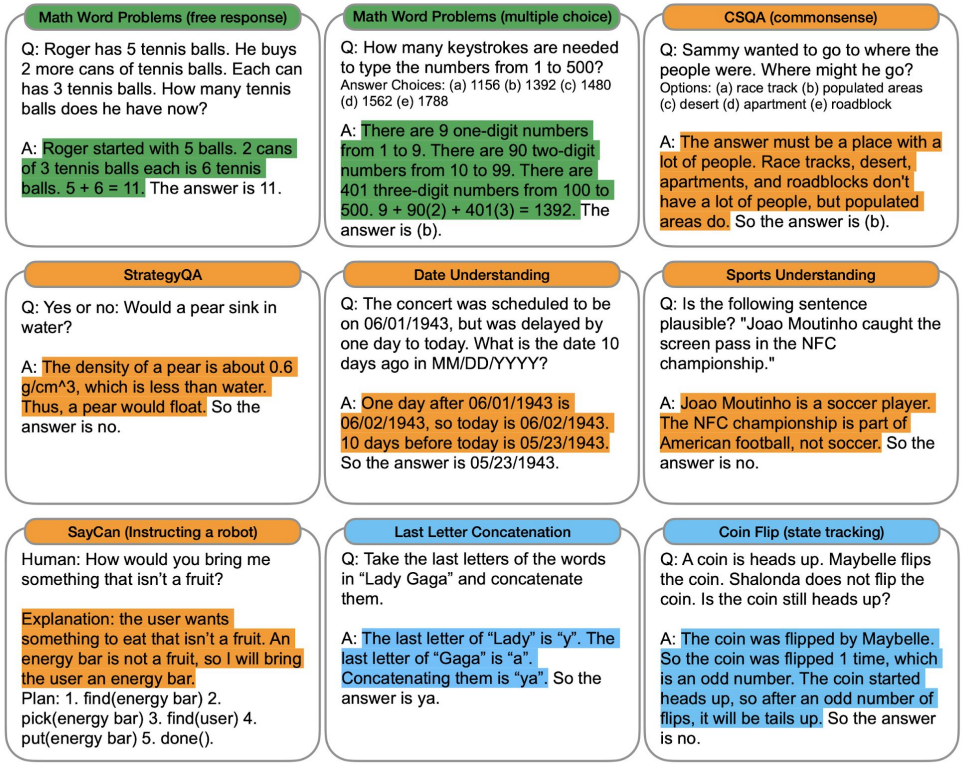


Figure 3: Examples of ⟨input, chain of thought, output⟩ triples for arithmetic, commonsense, and symbolic reasoning benchmarks. Chains of thought are highlighted. Full prompts in Appendix G.

https://arxiv.org/pdf/2201.11903

Meta Prompting (4/16)

Align model outputs with instructions more explicitly.

“Provide rules about how prompts should be interpreted.”

Category: Output Customization

“Whenever I say ‘expand’, provide a detailed explanation.”

“From now on, ‘a → b’ means a directed edge in a graph.”

“When I ask ‘check’, always verify facts before answering.”

“Interpret ‘compress’ as ‘summarize into one sentence.’”

“If I say ‘draft’, return a structured outline, not full prose.”

Self-Consistency (5/16)

Avoid brittle single-response reasoning.

“Sample multiple reasoning paths and select majority answer"

Category: Reliability

“Generate 5 different solutions to this puzzle, then vote for the best.”

“Write 3 alternative translations, then choose the most natural one.”

“Solve this math problem three ways and pick the consistent answer.”

“Give 4 candidate titles for this article and select the most fitting.”

“Provide 3 chains of reasoning for this riddle and decide which aligns.”

Generated Knowledge Prompting (6/16)

Model lacks context about a topic.

“First generate background knowledge, then solve task.”

Category: Input Expansion

“List key facts about climate change. Then, explain why sea levels rise.”

“Recall background about the Cold War. Then, analyze causes of Berlin Wall fall.”

“Generate relevant formulas for acceleration. Then, solve this physics problem.”

“First, outline historical context of the Renaissance. Then, describe its impact.”

“Provide scientific definitions of photosynthesis. Then, answer: why is it essential?”

Prompt Chaining (7/16)

Complex tasks requiring multiple stages.

“Break down task into smaller sequential prompts.”

Category: Orchestration

Step 1: Extract names from text. Step 2: Categorize them as ‘person’ or ‘place.’

Step 1: Summarize article. Step 2: Create 3 quiz questions.

Step 1: Generate story idea. Step 2: Write dialogue for the main character.

Step 1: Identify equations. Step 2: Solve them.

Step 1: Translate text to French. Step 2: Summarize it in French.

Tree of Thoughts (8/16)

Explore multiple solution paths systematically.

“Branch into different reasoning steps, evaluate, then choose.”

Category: Reasoning

“List 3 possible moves in this chess scenario. Explore each outcome. Choose best.”

“Propose 3 business strategies. Expand pros/cons. Pick optimal.”

“Suggest 4 interpretations of this poem. Expand reasoning. Select strongest.”

“Give 2 possible explanations for test failure. Expand, compare, decide.”

“Explore 3 ways to market this app. Evaluate and finalize.”

Automatic Reasoning (9/17)

Model alone can’t perform certain operations.

“Instruct model to invoke tools/APIs for tasks.”

Category: Orchestration

“If math → use calculator tool. Otherwise, explain in text.”

“Look up current weather via API. Then summarize.”

“Retrieve code snippet. Run it. Report output.”

“For numbers, use Python execution. For text, use natural explanation.”

“Query database for stock prices, then provide analysis.”

Automatic Prompt Engineering (10/16)

Manually designing prompts is inefficient.

“Ask model to generate improved prompts automatically.”

Category: Meta-prompting

“Generate 5 candidate prompts for text summarization.”

“Optimize this prompt for clarity: ‘Explain black holes.’”

“Propose improved prompts for better code explanations.”

“Refine my query into a structured instruction.”

“Suggest 3 alternative prompts to maximize creativity.”

Active-Prompt (11/16)

Model may fail on ambiguous queries.

“Actively generate clarifying sub-questions before answering.”

Category: Reliability

“Before answering, ask clarifying questions if ambiguous.”

“If user request unclear, propose 3 interpretations and ask which one.”

“Generate follow-up questions before giving final answer.”

“Identify missing information in query, then proceed.”

“Always confirm assumptions before completing task.”

Directional Stimulus Prompting (12/16)

Output drifts from desired direction.

“Provide guiding keywords or signals.”

Category: Semantics

“Summarize this article (focus on health impacts).”

“Explain this concept (highlight technical aspects).”

“Generate ideas (prioritize creativity).”

“Translate text (preserve humor style).”

“Write essay (emphasize environmental benefits).”

Program-Aided Language Models (13/16)

Model reasoning weaker than formal execution.

“Convert reasoning steps into code execution.”

Category: Orchestration

“Convert problem into Python code, run, then return answer.”

“Generate SQL query, execute, and provide result.”

“Write script to simulate dice rolls, then summarize.”

“Formulate function for Fibonacci sequence, then compute 10th term.”

“Encode logic in code, run it, then explain result.”

ReAct (14/16)

Need reasoning plus factual retrieval.

“Alternate between reasoning and acting.”

Category: Reasoning + Tool Use

“Reason: I need recent weather. Act: Query weather API. Reason: Summarize.”

“Reason: Solve math problem. Act: Use calculator. Reason: Confirm answer.”

“Reason: Identify fact gap. Act: Retrieve doc. Reason: Conclude.”

“Reason: Need citation. Act: Search source. Reason: Integrate.”

“Reason: Plan steps. Act: Execute tool for each. Reason: Summarize final.”

Reflexion (15/16)

Improve model outputs via self-feedback.

“Generate answer, critique it, then refine.”

Category: Reliability

“Draft answer. Critique it. Revise with corrections.”

“Solve problem. Reflect on errors. Provide corrected version.”

“Generate summary. Evaluate clarity. Rewrite improved.”

“Write code. Debug self. Return fixed code.”

“Answer question. Reflect on biases. Refine unbiased answer.”

Graph Prompting (16/16)

Need structured relational reasoning.

“Represent information as nodes and edges for the model.”

Category: Input Structuring

“Represent relationships as graph: A → B, B → C. Explain paths.”

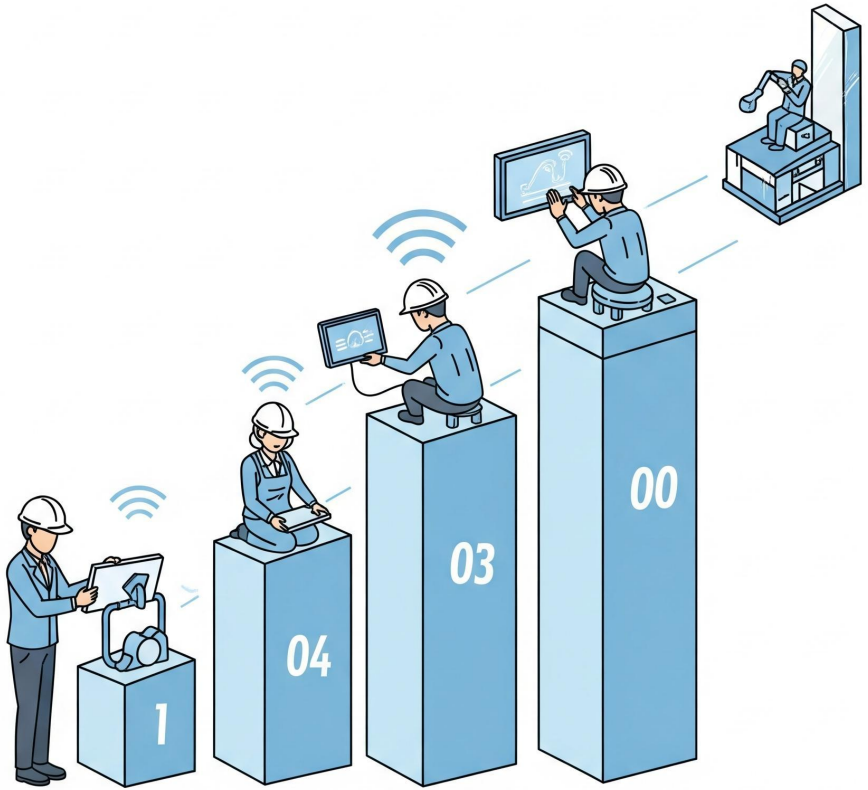
“Build knowledge graph from text. Summarize connections.”

“Show dependencies as nodes/edges. Identify bottlenecks.”

“Encode family tree as graph. Explain lineage.”

“Represent workflow as graph. Highlight cycles.”

Remember about…
Levels of Automation



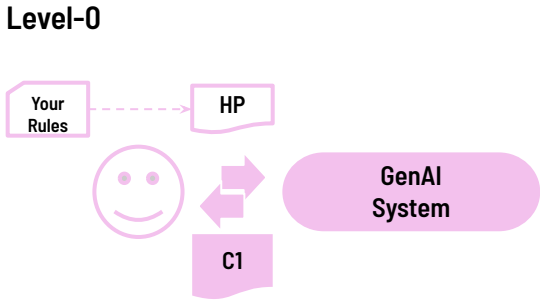
Level-0: GenAI as a QA Tool

Process:

"Provide me an overview about [problem/topic]"

Your response is as good as your ability to prompt.

- Understanding of the Problem/Question
- Knowledge for contextualization
- Capacity for conceptualization
- Writing Skills



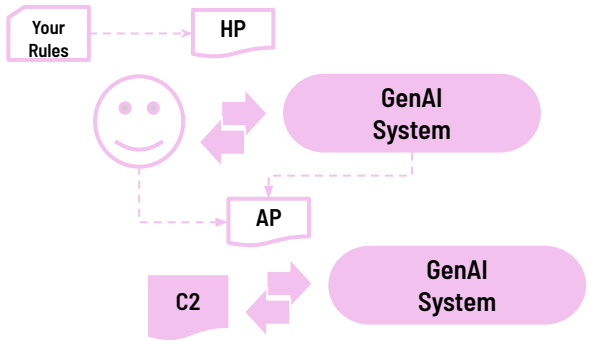
Level-1: GenAI creates the Question for GenAI

"Create a prompt to be used as instructions to a GPT to clear, intuitive response about [problem]".

Automated prompts' tend to yield better results than good human prompts?

- Metacognition.
- Prompt Engineering.
- Better contextualization.

Level-1



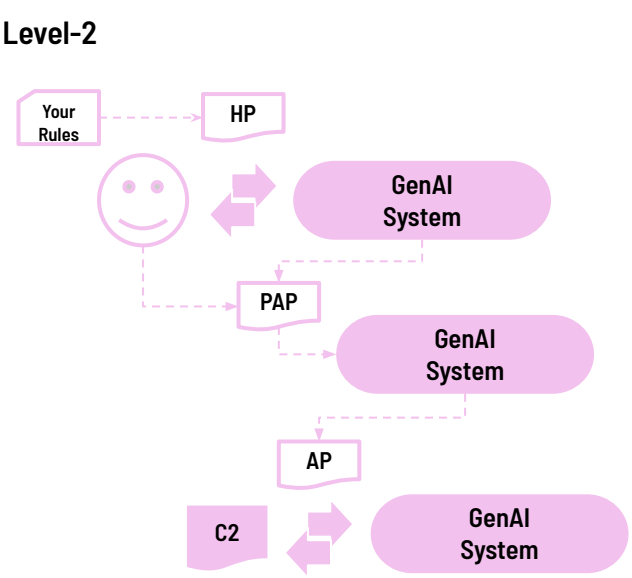
HP - Human Prompt
AP - Automated Prompt

Level-2: GenAI help to establish the 'rules' to create the Prompt requesting for a Prompt

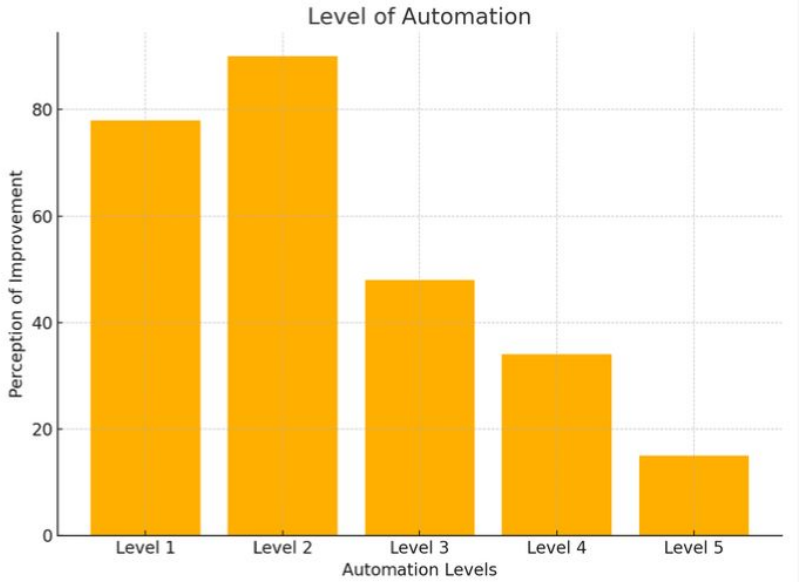
"Provide the strategy to explore [ideas|concepts] around [problem|topic] in preparation to write prompts to create instructions for a GPT that will provide [answers] for that [problem|topic]"

You are adding:

- Metacognition.
- Rule-driven Prompt engineering.
- Way Better contextualization.
- Critical Thinking.



You can keep going, but …
Higher-levels of automation (>2) negatively impact performance!



Exercises

Exercise 3 - Prompt Engineering Lab

Objective:
Create, refine, and evaluate various prompt strategies to achieve specific generation goals, comparing effectiveness across different model configurations.

Go to Canvas -> Assignments

COT 6930 - Generative Intelligence and Software Development Lifecycles

Dr. Fernando Koch
kochf@fau.edu
http://www.fernandokoch.me