Prompt Engineering & Modern AI Tooling

CONTENT

PROMPT ENGINEERING

What is Prompt Engineering?

- A technique to communicate with AI models (like ChatGPT) using natural language.
- You give the model a prompt (a question or instruction), and it generates a response.
- Acts like programming using words instead of code.

Why Prompt Engineering?

- Helps you get more accurate and useful answers from the model.
- Makes it easier to control or guide the AI toward a specific task.
- Used in many fields: chatbots, content creation, education, automation.

BETTER PROMPTS GIVE BETTER RESULTS...

PROMPTING TECHNIQUES

What Are Prompting Techniques?

Prompting techniques are different ways to ask questions or give instructions to an AI model to get better, more useful answers.

Common Prompting Techniques

1. Zero-Shot Prompting – Ask without giving examples.
2. Few-Shot Prompting – Include a few examples to guide the answer.
3. Chain-of-Thought (CoT) – Ask the model to reason step by step.
4. Role-Based Prompting – Assign the model a role or personality.

Why Prompting Techniques Matter

- Improve the quality of AI output.
- Help AI handle complex or open-ended tasks.
- Make your prompts more predictable and reusable.

ZERO-SHOT vs FEW-SHOT PROMPTING

What is Zero-Shot Prompting

- You give the model an instruction only, with no examples.
- The model uses its pretrained knowledge to respond.

```
Prompt: Translate to Sinhala: I am happy
Response: මම සතුටින් ඉන්නවා
```

What is Few-Shot Prompting

- You provide a few examples of the task in the prompt.
- The model uses the examples as a pattern to follow.

```
Prompt: Translate these:
        - Hello → හෙලෝ
        - Thank you → ඔබට ස්තූතියි
        - I am happy →
Response: මම සතුටින් ඉන්නවා
```

Feature	**Zero-Shot**	**Few-Shot**
Examples Given	No	Yes (2–5 examples)
Prompt Length	Short	Longer
Performance on Complex Tasks	Sometimes weak	Usually better

Why Use These

- Use Zero-Shot for easy or common tasks.
- Use Few-Shot when you need clear format, structure, or accuracy.

CHAIN-OF-THOUGHT & ROLE-BASED PROMPTING

Chain-of-Thought (CoT) Prompting

- Encourages the model to think step by step before giving the final answer.
- Useful for tasks that involve logic, math, or reasoning.

```
Prompt: A person has 3 apples and buys 2 more. How many apples now? Think step by step.
Response: The person had 3 apples. They bought 2 more. 3 + 2 = 5 apples.
```

Role-Based Prompting

- Ask the model to act as a specific expert or role.
- Useful for tailoring answers for a target audience or domain.

```
Prompt: You are a science teacher. Explain gravity to a 10-year-old.
Response: Gravity is the force that pulls everything down to the ground — like when you drop a ball.
```

Feature	**Chain-of-Thought Prompting**	**Role-Based Prompting**
Purpose	Helps the model reason step-by-step	Makes the model respond like a specific role/person
Use Case	Math, logic, multi-step tasks	Teaching, customer support, expert advice
Prompt Style	Ask model to "Think step by step"	Ask model to "Act as..." or "You are a..."

Why Use These

- CoT improves reasoning accuracy.
- Role-based prompts create more relevant and audience-aware answers.

PROMPT QUALITY – GOOD vs BAD EXAMPLES

Aspect	Bad Prompt	Good Prompt
Too vague	"Tell me about climate."	"Explain how climate change affects farming in 3 bullet points."
No context	"Summarize this."	"Summarize the following article for a high school student."
Overly broad	"Write something interesting."	"Write a short blog post about AI in education (100 words)."

Tips for Writing Better Prompts

- Be specific about what you want.
- Include the format (e.g., bullet points, paragraph, list).
- Give context or audience (e.g., for kids, for developers).
- Use clear instructions and desired tone.

CORE ELEMENTS OF A GOOD PROMPT

Let...

"You are a friendly history teacher. Summarize the following article about World War II, focusing on its causes, and answer in 3 simple bullet points using clear and easy-to-understand language. For example, you might write something like: "Germany invaded Poland in 1939, which started the war."

Element	Description	Example Snippe
Task	What do you want the model to do?	"Summarize", "Translate", "Classify", "Answer..."
Context	Background or extra information the model needs	"Based on the article below..."
Exemplars	One or more examples showing the desired pattern or output	"Translate: Hello → Hola, Thank you → Gracias..."
Persona	The role or identity the model should adopt	"You are a history teacher..."
Format	The structure or output format you expect	"Answer in 3 bullet points."
Tone	The style or emotion you want (formal, friendly, etc.)	"Explain in a casual and simple tone."

INTRODUCTION TO GENERATIVE AI

What is Generative AI?

A type of artificial intelligence that can create new content such as

- Text
- Images
- Code
- Music
- Video

How Does It Work?

1. Built using Large Language Models (LLMs) like GPT, Claude, Gemini, etc.
2. Learns from huge amounts of data (books, articles, websites) to predict and generate content.
3. Uses deep learning (especially transformers) to understand and generate human-like responses.

What Can It Do

- Write articles, emails, poems, essays
- Translate languages
- Answer questions
- Generate code
- Summarize documents
- Chat like a human

HOW GENERATIVE AI WORKS

Core Technology Behind Generative AI

- Powered by Large Language Models (LLMs) like GPT, BERT, Claude, etc.
- LLMs are trained on massive datasets (books, websites, code) to predict the next word or token.
- Use deep learning architectures called Transformers.

What is a Transformer?

- A neural network model designed to process and understand sequences of data (like text).
- Introduced in the paper "Attention is All You Need" (2017).
- Uses attention mechanisms to focus on important words in a sentence.

How the Model Works (Simplified Flow):

- Input: You give a prompt
  "Translate: I am happy"
- Tokenization: Text is broken into tokens (words or parts of words).
- Processing: The model predicts the next token based on patterns learned during training.
- Output: The final response is generated →
  "මම සතුටින් ඉන්නවා"

Where is Generative AI Used?

Domain	Application Example
Content Creation	Writing blogs, articles, emails, social media posts
Education	AI tutors, personalized learning, automated grading
Customer Support	AI chatbots, email response automation
Translation	Real-time language translation
Search Engines	Smart search with AI-generated summaries
Programming	Code generation, debugging, code explanation
Design & Art	AI image generation (e.g., DALL·E, Midjourney)
Media & Entertainment	Script writing, lyrics, storyboarding, video synthesis

Generative AI is not just for chatting . It's a powerful tool across industries.

LIMITATIONS OF GENERATIVE AI

1. Hallucination (False Information)
   - AI can generate content that sounds correct but is factually wrong.
   - Example: Giving fake references or making up historical facts.
2. Lack of True Understanding
   - AI doesn’t “understand” — it predicts based on patterns, not meaning.
   - It has no common sense or awareness of the real world.
3. Bias in Output
   - AI can reflect biases from training data (gender, race, politics).
   - May produce unfair or offensive responses.
4. Not Always Reliable for Logic Tasks
   - May struggle with math, reasoning, or multi-step instructions, especially without chain-of-thought prompts.
5. Privacy and Security Risks
   - AI can accidentally leak sensitive data if trained improperly.
   - Using private or unverified models can pose risks.

Generative AI is powerful, but not perfect. Always verify, guide, and use it responsibly.

WHY PROMPT ENGINEERING IS IMPORTANT IN GENAI

1. Directly Affects Output Quality
   - The same model can give bad or brilliant answers depending on the prompt.
   - Well-designed prompts lead to more accurate, useful, and safe results.
2. No Need to Change the Model
   - Prompt engineering lets you improve behavior without retraining the model.
   - It’s like adjusting the input instead of rewriting the software.
3. Unlocks Model Capabilities
   - Advanced techniques (e.g., CoT, role-based prompts) help models perform reasoning, summarization, translation, etc., better.
4. Essential for Real-World Use
   - In apps like chatbots, search, and content tools, good prompting is the main control method.
   - Companies like OpenAI, Google, and Meta use prompt optimization in production systems.

INTRODUCTION TO AGENTIC AI

WHAT IS AGENTIC AI?

Agentic AI refers to AI systems that can autonomously plan, decide, and act across multiple steps to complete a goal like an intelligent assistant that thinks, plans, and executes.

How It's Different from Regular AI

- Traditional AI = Responds to single prompts (one-shot).
- Agentic AI = Can reason, take actions, use tools, and make decisions in a loop.

Core Abilities of Agentic AI

Ability	**Description**
Perceive	Understand a prompt or environment
Plan	Break the task into steps
Act	Use tools or make API calls
Reflect	Check if progress is made or adjust the plan
Repeat	Loop until task is completed

Real-Life Analogy

Like giving an intern a task:
"Find 5 hotels near Colombo, check reviews, and send me the best option."
The intern (or AI agent) breaks this into subtasks and does it all — not just answers one question.

HOW AGENTIC AI WORKS

Agentic AI Workflow

Stage	Description
Think	Understand the user’s goal or query
Plan	Break the goal into a sequence of smaller steps
Act	Take actions like calling APIs, using tools, or accessing memory/files
Reflect	Check if progress is correct; revise steps if needed
Repeat	Continue until the final result is reached

Example: Booking a Trip (AI Agent Task)

Goal: “Book me a weekend trip to Nuwara Eliya with a hotel and train schedule.”
Agent's internal process:

- Think: Understand it's a travel booking task
- Plan: Find train times → Search hotels → Compare prices
- Act: Call train schedule API, hotel API
- Reflect: Choose best-rated hotel, adjust if unavailable
- Complete: Return full itinerary

AGENTS vs PROMPTS – KEY DIFFERENCES

Basic Comparison Table

Feature	Prompt-Based AI	Agentic AI
Behavior	Responds to a single prompt	Handles a full task with multiple steps
Control	Human writes and guides every prompt	Agent plans and executes autonomously
Memory	Usually stateless (forgets past prompts)	Maintains state and memory across steps
Tool Use	Limited or none	Can use tools, APIs, databases, and web access
Iteration	One prompt → one response	Can reflect, retry, and refine responses
Best For	Simple Q&A, summaries, explanations	Complex goals, workflows, automation

Simple Analogy

- Prompt-based AI is like asking a search engine one question at a time.
- Agentic AI is like hiring an assistant who does the entire job for you — planning, acting, and reporting back.

The agent did multiple actions, used tools, and made decisions — without needing more user input.

GENERATIVE AI vs AGENTIC AI – COMPARISON

Basic Comparison Table

Feature	Generative AI	Agentic AI
Interaction Style	Responds to single prompts	Handles multi-step tasks autonomously
Intelligence Type	Predictive (based on pattern completion)	Goal-oriented (plans, acts, reflects)
Task Handling	One task at a time	Breaks down complex tasks into steps
Tool Use	Typically none or limited	Can use tools, APIs, memory, and external resources
Memory & Context	Short-term memory (limited session context)	Long-term memory, stateful reasoning
Best For	Text generation, Q&A, summarization	Research, automation, workflows, multi-step reasoning
Example	"Summarize this article"	"Find 3 sources, compare them, and give the best one"

Summary

- Generative AI gives you answers,
- Agentic AI gives you solutions.

MOVING FROM PROMPTS TO AGENTS

Why We Need Tools Beyond Prompting

Prompts alone are not enough for real-world, multi-step tasks. Agents need to:

- Maintain memory
- Plan and adapt
- Use tools (APIs, web search, files)
- Store intermediate steps and results

How the Model Works (Simplified Flow):

Without Tools	With Tools (Agent Frameworks)
One-shot interactions	Continuous, multi-step workflows
No external actions	Can use tools like search, database, APIs
Stateless	Maintains memory across steps
Only text output	Can return structured data, trigger real

We're moving from "talking to AI" to "working with AI agents."

RETRIEVAL-AUGMENTED GENERATION (RAG)

What is RAG?

RAG stands for Retrieval-Augmented Generation – a technique where the AI model first retrieves relevant information from a knowledge source, and then generates a response based on it.

How It Works (Simplified Flow)

- User Prompt →
  "Who is the current president of Sri Lanka?"
- Retrieve →
  Search in a document store, website, or knowledge base
- Generate →
  Use the retrieved info to craft an accurate response

Why Use RAG

Feature	Benefit
Up-to-Date Info	Can include facts not in the model’s training data
Long-Term Memory	Retrieves from custom data sources (PDFs, websites, DBs)
Accurate Output	Reduces hallucinations by grounding responses in real data

HOW RAG WORKS & WHEN TO USE IT

Architecture of RAG

Module	Role
Retriever	Finds relevant content from documents or knowledge base
Generator	Uses a language model (LLM) to produce a grounded response
Data Store	Vector database (like FAISS, Pinecone, Chroma) to store texts
Pipeline	Connects query → search → generation → response

When Should You Use RAG

Situation	Why RAG is Ideal
You have domain-specific documents	RAG can use your exact PDFs, notes, or manuals
Need to reduce hallucinations	Grounding improves factual accuracy
Want LLM access to updated info	Without retraining, just update the knowledge source
Require transparent and explainable output	You can show exactly where the answer came from

Quick Example

Ask: “List holidays in Sri Lanka in 2025”

LLM searches a live holiday calendar → returns accurate list with sources.

LANGCHAIN & THE LANG ECOSYSTEM

What is the Lang Ecosystem?

- The Lang ecosystem is a set of tools built around LangChain to help you design, run, debug, and manage advanced LLM applications.
- Each tool handles a different part of the AI development lifecycle.

Lang Ecosystem Overview

1. LangChain
2. LangGraph
3. LangFlow
4. LangSmith

Why It Matters

- You don’t need to build everything from scratch.
- The Lang ecosystem modularizes LLM app development, making it faster, safer, and scalable.
- Great for building real-world GenAI agents with tools, memory, and logic.

INTRODUCTION TO LANGCHAIN

What is LangChain?

LangChain is a framework for building LLM-powered applications that can:

- Use prompts
- Interact with external tools and APIs
- Store and retrieve memory
- Handle multi-step workflows

It turns a simple chatbot into a powerful agent system.

Why Use LangChain?

- Simplifies building complex AI apps
- Enables agentic behavior using prompt + tools + memory
- Supports chatbots, assistants, automation tools, and more

Real Example Use Case:

A chatbot that answers university policy questions by reading your uploaded PDFs → Built using LangChain + RAG

LANGCHAIN ARCHITECTURE & COMPONENTS

High-Level Architecture Overview

[User Input] ⟶ [Prompt Template] ⟶ [Chain / Agent] ⟶ [LLM + Tools + Memory + Retriever] ⟶ [Response]

Key Components

Component	Purpose
PromptTemplate	Template for structured, reusable prompts
Chain	Sequence of steps (LLM → tool → format → response)
Agent	Dynamic decision-maker that can choose what to do next
Memory	Stores history or conversation context
Tool	External actions (e.g., web search, calculator, API)
Retriever	Retrieves relevant documents for RAG

How It Works Together:

1. User asks a question
2. Chain formats the prompt → LLM is called
3. If needed, the agent decides to use a tool
4. Memory + retriever enhance the output
5. Final result is returned to the user

INTRODUCTION TO LANGGRAPH

What is LangGraph?

LangGraph is a library built on top of LangChain that lets you create multi-step, stateful AI agents using graph-based workflows.

Key Idea

Instead of chaining steps in a straight line (like LangChain), LangGraph uses a graph – where each node represents a step, and edges define how control flows.

Why Use LangGraph?

Feature	Benefit
Loops & Branching	Supports cycles, retries, and conditional flows
Stateful Agents	Maintains memory and state at each decision point
Visual Workflow	Easy to visualize and debug agent behavior
Asynchronous Logic	Good for real-time and complex background operations

How It Relates to LangChain

- LangChain helps define logic using chains and agents
- LangGraph structures that logic into a graph of states and decisions

LANGGRAPH ARCHITECTURE & EXAMPLE FLOW

LangGraph Architecture Overview

LangGraph builds a state machine where:

- Nodes = Functions or LLM steps
- Edges = Conditions or decisions that define next step
- State = Memory or context passed through the graph

Basic Flow

[Start] ⟶ [Input Handler Node] ⟶ [LLM Node] ⟶ [Tool Use Node] ⟶ [Reflect or Retry Node] ⟶ [Final Answer Node]

Real Example: Research Agent

Goal: "Summarize the top 3 recent papers on AI ethics."

Graph Nodes:

- Search Papers
- Filter by Date & Relevance
- Summarize Abstracts
- Check Quality
- Return Final Summary

Loops if summaries are missing or irrelevant

INTRODUCTION TO LANGFLOW

What is LangFlow?

LangFlow is a visual drag-and-drop interface for building and testing LangChain applications, no coding required.

Purpose

- Makes it easy to design, visualize, and debug chains and agents
- Great for prototyping, learning, and non-programmers

Key Features

Feature	What It Offers
Node-based UI	Drag-and-drop components (LLMs, prompts, tools, memory, etc.)
Live Testing	Run your pipeline and see outputs immediately
Chain Design	Connect prompts, logic, and tools visually
Export Options	Save flows and generate backend code

Use Case

A student builds a custom AI assistant that reads PDFs and answers questions all visually, using LangFlow.

Example LangFlow Pipeline

What Happens in a LangFlow Pipeline

LangFlow allows you to visually design the logic of your AI application by connecting building blocks (called “nodes”), each responsible for a specific task.

Basic Flow

[Input Node] ⟶ [IPrompt Template Node] ⟶ [LLM Node] ⟶ [Tool Use Node] ⟶ [Memory Node] ⟶ [Output Node]

Example Scenario

User Asks: “Explain the attendance rules from the handbook.”

- LangFlow retrieves the matching section from the PDF
- LLM reads it and summarizes
- Answer: “You must attend 80% of lectures to sit for final exams.”

Why This Matters

- Students can build working AI tools visually
- Helps understand how components (LLM, RAG, memory) interact
- No advanced coding required. Just logic and design

INTRODUCTION TO LANGSMITH

What is LangSmith?

- LangSmith is a platform that helps debug, evaluate, and monitor your LLM applications built with LangChain (or similar frameworks).
- It acts like a lab environment to improve prompt workflows, agents, and tool usage.

What Can You Do with LangSmith

Feature	Purpose
Trace Chains	See the full call flow — prompts, inputs, outputs, retries
Debug Agents	Inspect decisions, tool usage, memory state
Evaluate Quality	Run tests on outputs (accuracy, relevance, consistency)
Store Runs	Save and compare runs over time
Prompt Testing	Try different prompts and versions in a controlled environment

Why LangSmith Matters

- Helps you understand why the AI is behaving a certain way
- Essential for iterating, improving, and deploying safely

How LangSmith Helps with Testing and Debugging

Debugging with LangSmith

1. Trace Chains & Agents:
   View the entire flow — from prompt input to final response.
2. Visual Prompt Debugging:
   Inspect input variables, tokens, tools used, and intermediate outputs.
3. Version Comparison:
   Track how a change in prompt/template impacts accuracy or tone.
4. Run Evaluation Metrics:
   Automate tests using criteria like exact match, BLEU, ROUGE, or custom scoring.

Why It Matters

- Fix errors faster
- Improve prompt quality
- Validate agent behavior before deployment

Thank You..