< CodeVigyaan />

Prompt Engineering

- Introduction
  - LLMs and how they work?
  - What is a Prompt?
  - What is Prompt Engineering?
  - Models offered by _____
    - OpenAI
    - Google
    - Anthropic
    - Meta
    - xAI
- Common Terminology
  - LLM
  - Tokens
  - Context Window
  - Hallucination
  - Agents
  - Prompt Injection
  - Model Weights / Parameters
  - Fine-Tuning vs Prompt Engg.
  - AI vs AGI
  - RAG

LLM Configuration

- Sampling Parameters
  - Temperature
  - Top-K
  - Top-P
- Output Control
  - Max Tokens
  - Stop Sequences
- Repetition Penalties
  - Frequency Penalty
  - Presence Penalty
- Structured Outputs

Prompting Techniques

- Zero-Shot Prompting
- One-Shot / Few-Shot Prompting
- System / Role / Contextual
  - System Prompting
  - Role Prompting
  - Contextual Prompting
- Step-back Prompting
- Chain of Thought (CoT) Prompting
- Self-Consistency Prompting
- Tree of Thoughts (ToT) Prompting
- ReAct Prompting
- Automatic Prompt Engineering
  - Use LLM to generate Prompts
- AI Red Teaming
  - AI Red Teaming Roadmap

Prompting Best Practices

— Provide few-shot examples for structure or output style you need

— Keep your prompts short and concise

— Ask for structured output if it helps e.g. JSON, XML, Markdown, CSV etc

— Use variables / placeholders in your prompts for easier configuration

— Prioritize giving clearer instructions over adding constraints

— Control the maximum output length

— Experiment with input formats and writing styles

— Tune sampling (temperature, top-k, top-p) for determinism vs creativity

— Guard against prompt injection; sanitize user text

— Automate evaluation; integrate unit tests for outputs

— Document and track prompt versions

— Optimize for latency & cost in production pipelines

— Document decisions, failures, and learnings for future devs

— Delimit different sections with triple backticks or XML tags

Improving Reliability

- Prompt Debiasing
- Prompt Ensembling
- LLM Self Evaluation
- Calibrating LLMs

https://codevigyaan.com/