# 1. Prompting and Its Types

---

### 1.1 Introduction

Prompting is the process of giving instructions or queries to a **Large Language Model (LLM)** like GPT or Gemini to get a desired output.
In simple terms, a *prompt* is the input text that guides the model to perform a specific task such as text generation, translation, summarization, or question answering.

**Why it is used:**

- It helps communicate user intent to the model.
- The quality of the output depends heavily on how well the prompt is designed.
- Effective prompting improves **accuracy, coherence, and creativity** of AI responses.

---

### 1.2 Core Concepts

#### 1.2.1 What is a Prompt?

A prompt can be a **question, instruction, or context** given to an AI model to produce a relevant response.

**Example:**

```
from openai import OpenAI

client = OpenAI()


response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[{"role": "user", "content": "Write a short poem about the ocean."}]
)
print(response.choices[0].message.content)
```

**Output:**

The ocean whispers secrets deep,

Where coral dreams and shadows sleep…

---

### 1.3 Types of Prompting

#### 1.3.1 Zero-Shot Prompting

- No example is provided.
- The model relies solely on its pre-trained knowledge.

**Example:**

Prompt: Classify the sentiment of this sentence: "I love this movie!"

**Output:**

Positive

---

#### 1.3.2 One-Shot Prompting

- One example is provided to guide the model.

**Example:**

Prompt:

Example: "The food is great!" → Positive

Now classify: "The service was terrible."

**Output:**

Negative

---

#### 1.3.3 Few-Shot Prompting

- A few examples (2–5) are provided to help the model understand the pattern.

**Example:**

Prompt:

"The weather is nice." → Positive

"I hate traffic jams." → Negative

"The view is beautiful." → Positive

Now classify: "The food is cold."

**Output:**

Negative

---

### 1.3.4 Chain-of-Thought (CoT) Prompting

- Encourages the model to “think step-by-step.”
- Useful for reasoning or logical problems.

**Example:**

Prompt:

If a car travels 60 km in 2 hours, what is its average speed?

Let's think step by step.

**Output:**

Step 1: Distance = 60 km

Step 2: Time = 2 hours

Step 3: Speed = Distance / Time = 60 / 2 = 30 km/h

Answer: 30 km/h

---

### 1.4 Use Cases / Applications

- Chatbots and virtual assistants
- Automated email generation
- Sentiment analysis
- Translation and summarization
- Code generation and debugging

---

### 1.5 Related Tools / Integrations

- **LangChain** – to structure and chain prompts dynamically.
- **PromptLayer** – to track and optimize prompt performance.
- **OpenAI API / Hugging Face** – to send prompts to LLMs.

---

### 1.6 Conclusion

Prompting is the foundation of all LLM-based applications.
By choosing the right **type of prompt**, developers can guide models more effectively, leading to improved **accuracy, control, and interpretability**.

---

# 2. Prompt Tuning

---

### 2.1 Introduction

Prompt Tuning is an advanced technique used to **optimize prompts automatically** instead of writing them manually.
It involves training **special tokens (soft prompts)** that help the model perform better on a specific task without fine-tuning the whole model.

**Why it is used:**

- Saves computational resources compared to full model training.
- Improves model performance on **domain-specific tasks**.
- Makes LLMs more adaptable and efficient.

---

### 2.2 Core Concepts / Components

1. **Soft Prompts** – Learnable embeddings instead of text-based instructions.
2. **Task-Specific Adaptation** – Model learns the best prompt tokens for a given dataset.
3. **Frozen Model Weights** – Only the prompt embeddings are trained, not the entire model.
4. **Efficient Tuning** – Reduces training cost and data requirements.

---

**Example (Simplified Illustration)**

Suppose we want to classify reviews as positive or negative.

**Traditional Prompt:**

Prompt: Classify the sentiment of this review: "The product is amazing."

**Prompt-Tuned Version (Conceptually):**

<SoftPrompt> The product is amazing.

Here, <SoftPrompt> is a learned vector that improves classification accuracy.

---

**2.3 Diagram: Prompt Tuning Process**

[ Original Prompt ]

↓

[ Evaluation / Feedback ]

↓

[ Optimization / Tuning ]

↓

[ Improved Prompt ]

↓

[ Better Model Output ]

---

**2.4 Use Cases / Applications**

- Domain-specific tasks like **medical diagnosis** or **legal document analysis**
- **Customer support bots** trained on specific brand tone
- **Text classification**, **translation**, and **Q&A** tasks
- **Personalized AI assistants** with contextual tuning

---

**2.5 Related Tools / Integrations**

- **OpenAI Fine-tuning API** – enables prompt and model fine-tuning.
- **Hugging Face Transformers** – supports prompt tuning with adapters.
- **LangChain + PromptLayer** – helps experiment, log, and optimize prompt versions.

---

### 2.6 Conclusion

Prompt tuning bridges the gap between manual prompt design and full model training. It enables efficient **task adaptation**, **performance improvement**, and **cost savings**—making it a powerful tool for advanced LLM applications.