Prompt Engineering: MindBridge

All what you should know about Prompt Engineering:

Prompts are the textual inputs that you enter into your Large Language Models(LLMs)- such as ChatGPT, Gemini, Bing… -. The model then predicts an appropriate response to the prompt you entered.

Prompt Structure:

A prompt can contain any of the following components:

- Instruction: a specific task or instruction you want the model to perform.
- Context: can involve additional context that can steer the model to better responses.
- Input Data: is the question we are interested in finding a response for.
- Output Indicator: indicates the type or format of the output.

In the context of LLMs, prompt engineering is the process of structuring text that can be interpreted and understood by the model.

Best Practices:

- Be clear and concise
- Provide context
- Define a clear task
- Avoid open-ended questions
- Use keywords
- Iterate

Prompt Techniques:

1. Zero-Shot Prompting: It is the process of prompting, or giving a task without prior examples where the model will depend exclusively on its own training data.
2. Few-Shots Prompting:It is simple as Zero-shot prompting, but you give examples before giving instructions, such as steps.
3. Chain-of-Thoughts Prompting: Same as the ones before but you add the phrase: **“Let’s think step by step”**

Prompt Engineering in MindBridge Project

MindBridge integrates three neural interaction paradigms — **Cognitive-to-Linguistic**, **Emotionally Adaptive**, and **Brain-Controlled** — into a single, adaptive prompt-engineering system that controls ChatGPT. The system converts EEG-derived cognitive intent and affective state into structured prompts and control actions. This document defines the prompt templates, data flow, decision rules, evaluation metrics, and recommended datasets for model training and validation.

Objectives

- Convert detected brain intents into textual prompts reliably enough to demonstrate hands-free chat control.
- Adapt ChatGPT's response tone and complexity in real time to the user's affective state.
- Provide a robust brain-control channel for chassis-level commands (send, cancel, regenerate) using event-detection (SSVEP/P300) or simulated triggers where needed.

System Overview

(Refer to integrated architecture table in repository; summarizes acquisition → preprocessing → feature extraction → decoders → fusion → prompt controller → ChatGPT → UI.)

Prompt Engineering Strategy

1. **Two-Layer Prompting**

System (affective) layer: Global behavior instructions derived from the affective decoder. Example:

```
[SYSTEM]
You are an empathetic, concise, patient assistant. Use short
paragraphs when the user is stressed. Ask if clarification is needed.
```

-

User (intent) layer: Specific task instruction derived from cognitive intent mapping:

```
[USER]
Summarize the content below in 2-3 sentences: {context}
```

    -
2. **Confidence-based control**
    - Use classifier confidence thresholds (e.g., >0.75) to commit actions; between 0.4–0.75 require confirm; <0.4 trigger clarification.
3. **Affective adaptation policy**
    - Map emotion → `system prompt` modifications (tone, verbosity, structure). Maintain a temporal smoothing window (e.g., 10s) to avoid flip-flopping.
4. **Audit & Safety**
    - Every brain-initiated action adds a signed metadata object: `{source: "neuro", intent: "summarize", confidence: 0.82, emotion: "stressed"}`. This permits post-hoc review.

Prompt Templates (Canonical)

- `SUMMARIZE`: `"You are an expert assistant. Summarize the following content concisely (2-3 sentences): {context}"`
- `EXPLAIN_MORE`: `"Explain the following with examples for a {audience}: {context}"`
- `QUESTION_ANSWER`: `"Answer the user question concisely. Provide references and a next step if applicable: {question}"`
- `CLARIFY`: `"I detected low confidence in the input. Confirm to proceed: yes/no."`

Data Flow & Decision Logic

- **EEG** → **features** → *Cognitive Model* → `intent, confidence`
- **EEG** → **affect features** → *Affective Model* → `emotion, prob`

- **Fusion** → if `confidence > threshold && event not control` → compose final prompt: `system_prompt + user_prompt` and call ChatGPT.
- **If control event (SSVEP/P300)** → execute mapped UI action (send/cancel/regenerate).

Evaluation Protocol

- **Emotion model**: accuracy, F1, confusion matrix on test subjects (LOSO recommended).
- **Intent model**: classification accuracy on motor imagery tasks, information transfer rate (ITR) for BCI control.
- **End-to-End**: task success rate and command latency when using combined pipeline in demo. Collect subjective metrics: perceived responsiveness and comfort via short surveys.

Hackathon Deliverables

- Working Streamlit demo showing: real-time EEG visualization, detected intent + emotion, assembled prompt (visible), ChatGPT response, control actions.
- Code repo with training scripts, inference pipeline, Streamlit app, README, and sample preprocessed data.
- Short demo video (120s) for presentation.
