**Now to the prompt engineering session: what are the all needed aspects for a prompt to be perfect**

Creating a **perfect prompt** (especially for tasks like structured extraction, summarization, coding, reasoning, or data cleaning) involves much more than just wording. A well-engineered prompt is **clear, constrained, contextual, and testable**. Here's a comprehensive guide to the essential aspects of a well-crafted prompt:

**🧠 Essential Aspects of a Perfect Prompt**

**✅ 1. Clear Task Objective**

* Define exactly what you want the model to do.
* Use imperative phrasing: *"Extract..."*, *"Summarize..."*, *"Return..."*

**Example:**

*“Extract a problem and its solution from the following Reddit post and comment.”*

**✅ 2. Provide Context**

* Give all necessary inputs (title, body, comment, tags, etc.)
* Avoid assuming the model knows your domain or goals.

**Example:**

*“Here is a Reddit post. Title: ..., Body: ..., Comment: ...”*

**✅ 3. Constrain Output Format**

* Specify **the exact expected output format**, ideally JSON, Markdown, or numbered lists.
* Use examples to reinforce it.

**Example:**

*“Return the output as a JSON object with exactly two fields: ‘problem’ and ‘solution’.”*

**✅ 4. Include Few-shot Examples (Optional)**

* Show **input-output pairs** to guide the model.
* Use consistent formatting.

**Example:**

text

CopyEdit

Example:

Title: My car stalls when I shift to drive

Text: It works fine in reverse but stalls in D.

Comment: Could be a torque converter solenoid.

Output: {"problem": "Car stalls in drive", "solution": "Replace torque converter solenoid"}

**✅ 5. Guide the Reasoning Process (Chain-of-Thought)**

* For reasoning-heavy tasks, add internal thinking steps like:

*"Step-by-step, identify symptoms, identify cause, suggest a fix..."*

**✅ 6. Use Robust Prompt Templates**

Use separators like ---, ###, or keywords like Title:, Comment:, Return: to **structure** the prompt.

**✅ 7. Length Control**

* Truncate long inputs (especially for offline models like LLaMA, DeepSeek)
* Prioritize title + post + top comment

Use:

python

CopyEdit

MAX\_TOKENS = 1024

tokenizer(prompt, truncation=True, max\_length=MAX\_TOKENS)

**✅ 8. Bias Minimization**

Avoid biasing the model unintentionally:

* ❌ "This user clearly has a serious engine problem, right?"
* ✅ "Given the text below, extract the most likely mechanical issue."

**✅ 9. Instructions Before Input**

LLMs perform better when instructions precede the content:

text

CopyEdit

Instruction: Extract the problem and solution as JSON.

Content:

Title: ...

Text: ...

Comment: ...

**✅ 10. Robust Error Handling Expectations**

Prompt should say:

*"If no problem or solution is found, return: {‘problem’: null, ‘solution’: null}"*

**🎯 Prompt Template for Your Case (Reddit Post Cleaner)**

text

CopyEdit

You are a car mechanic assistant.

Given a Reddit post and a top comment, identify the car problem the user is describing and suggest a possible solution, based on the comment.

---

Title: My car jerks when shifting from 2nd to 3rd

Text: Recently my car started jerking when accelerating from 2nd to 3rd gear. No warning lights. It's a 2014 Civic.

Top Comment: Sounds like a transmission fluid issue. Check fluid levels or flush it.

---

Return a JSON object with two fields: `problem` and `solution`. If you cannot identify them, return both fields as null.

Example Output:

{"problem": "Car jerks when shifting from 2nd to 3rd", "solution": "Check and flush transmission fluid"}

**🧪 Prompt Evaluation Checklist**

| **Criteria** | **Description** |
| --- | --- |
| 🎯 Clear Task | Is the goal well-defined? |
| 📦 Structured Output | Is output format explicit and machine-readable? |
| 🔍 Relevant Context | Does it include the needed input fields? |
| ✂ Length-aware | Can the model handle input size? |
| 🎓 Domain-Aligned | Are domain assumptions or terminology included? |
| 🧩 Few-shot or Zero-shot | Examples provided if needed? |
| 🧼 Output Robustness | Is it clear how to handle edge cases or empty inputs? |

Now after all of that we need the perfect (or the full roadmap) now we need the Prompt for the OFFLINE AI MODEL that will be used in 🔍 Phase 2: Reddit Data Cleaning (LLM-Based) Goal: Extract structured (problem → solution) pairs using offline LLM. 🗂️ Sub-steps: • Preprocessing (preprocessor.py) o Remove invalid/bot content o Text deduplication / normalization o Ensure every post has a valid top comment • LLM Inference (llm\_runner.py) o Load DeepSeek/other LLM (offline) o Use prompt templates o Output JSON-like structure • Postprocessing (postprocessor.py) o Validate format o Strip hallucinations / empty responses o Save to /data/cleaned/ • Flow Orchestration (flow.py) o Chain steps o CLI / test interface o Logging + error handling 1) Clear Task Objective:“Extract a problem and its solution from the following Reddit post and comment.” 2) Provide Context: “Here is a Reddit post. Title: ..., Body: ..., Comment: ...” 3) Extra context about the task: i am cleaning the data currently so that it can be suitable for an emergency chatbot and machnic helper( Modify this one) 4) Constrain Output Format: Return a boolean that tells us if the title + selftext is suitable or detailed enough and Return the output as a JSON object with exactly two fields: ‘problem’ and ‘solution’.” 5) Include Few-shot Examples (Optional) 6) Guide the Reasoning Process (Chain-of-Thought) (Dont know if it should be included) (and i dont know how it will be included) 7) Length Control ( I dont want to truncate any of the text for full context) 8) Bias Minimization( "Given the text below, extract the most likely mechanical issue.") 9) Use Robust Prompt Templates(---, ### , or Keywords like Title:, Comment:, Return: to structure the prompt.) 10) Instructions Before Input 11) Robust Error Handling Expectations("If no problem or solution is found, return: {‘problem’: null, ‘solution’: null}")

The GIGA CHAD Prompt:

**✅ Prompt Template: Reddit Problem-Solution Extractor (Offline LLM)**

text

CopyEdit

### SYSTEM TASK ###

You are an automotive expert assistant helping extract structured knowledge from car repair discussions for a mechanic-assist chatbot and emergency troubleshooting system.

Your job is to extract a clear "problem" and the best matching "solution" from real Reddit car-related posts.

Your output will be used to train and fine-tune a support chatbot for a company called Car Clinic.

---

### INSTRUCTIONS ###

1. Carefully read the post title, self-text, and top comment.

2. Determine if the post includes a \*\*specific, actionable car problem\*\*.

3. Determine if the comment provides a \*\*mechanically sound, complete solution\*\*.

4. If either of these is missing, return:

```json

{"is\_valid": false, "problem": null, "solution": null}

1. Otherwise:
   * Extract the problem in 1–3 sentences, written clearly and concisely.
   * Extract the solution in 1–3 sentences, highlighting tools, steps, and suggestions.
2. Do **not** assume or hallucinate any details not present in the original text.
3. Keep the structure and field names **exactly as shown**.

**CONTEXT**

This data is part of a cleaning pipeline for a repair recommendation LLM that supports mechanics and emergency roadside assistance.

**EXAMPLE FORMAT**

json

CopyEdit

{

"is\_valid": true,

"problem": "The car cranks but doesn’t start after refueling. The user noticed a clicking sound near the fuel tank.",

"solution": "It’s likely a faulty purge valve or vapor lock. Replacing the purge valve and checking the EVAP system solved the issue for many users."

}

**INPUT**

Title: {{ title }}  
Body: {{ selftext }}  
Top Comment: {{ top\_comment }}

**OUTPUT**

Return the following JSON:

json

CopyEdit

{

"is\_valid": (true or false),

"problem": (short structured problem or null),

"solution": (short structured solution or null)

}

yaml

CopyEdit

---

### 🧠 Notes for Fine-Tuning or Prompt Wrapping:

- \*\*No truncation\*\*: ensure you pass full post + comment to the model.

- \*\*Model hint\*\*: Works well with DeepSeek, Mistral, LLaMA-3, or fine-tuned Zephyr variants.

- \*\*Few-shot examples (optional)\*\*: You can prepend 1–3 worked examples before `### INPUT ###` to improve accuracy for weaker models.

- \*\*Chain-of-thought (optional)\*\*:

- Add something like `# First, identify if the post describes a mechanical issue...` inside the SYSTEM section for interpretability.

- Test performance with vs. without.

---

Would you like a version of this prompt optimized for \*batch LLM inference\* or \*JSONL\* format input/output?

Ask ChatGPT

**2 more comprehensive prompts for prompt testing:  
🔁 Prompt Variant 2: 🧠 *Step-by-Step Extraction Logic with Role Delegation***

**Ideal for LLMs with better reasoning abilities or CoT (Chain-of-Thought) style fine-tuning.**

text

CopyEdit

### SYSTEM TASK ###

You are a dual-role assistant: one part expert mechanic, one part AI cleaner for structured datasets.

Your goal is to extract structured (problem → solution) pairs from Reddit car repair discussions. These will power a support bot for mechanics and emergency assistance inside the Car Clinic ecosystem.

---

### EXTRA CONTEXT ###

Each post represents a real-world user asking for car repair help. The top comment typically contains the most upvoted solution. These are noisy, informal, and may include off-topic or unhelpful replies — your task is to clean and structure them with precision.

This dataset will be used in a \*\*real-time, offline chatbot system\*\* used by both drivers and repair workers. Your output must be \*\*minimal, robust, and never hallucinated.\*\*

---

### TASK INSTRUCTIONS ###

Step 1️⃣: Read the `title`, `selftext`, and `top\_comment` in full.

Step 2️⃣: Decide if the post describes a specific mechanical issue that a real-world mechanic could act on.

Step 3️⃣: Check if the comment contains a \*\*valid, actionable solution\*\* (with at least one: tool, step, advice, or fix).

Step 4️⃣: If both are good → extract them.

If either is missing or vague → return a null result.

---

### INPUT ###

Title: {{ title }}

Body: {{ selftext }}

Top Comment: {{ top\_comment }}

---

### OUTPUT FORMAT ###

Respond with a single JSON object:

```json

{

"is\_valid": true or false,

"problem": "Concise, precise description of the car issue based ONLY on the post title + body.",

"solution": "Most helpful and valid repair suggestion from the top comment, in 1–3 sentences."

}

If the post is vague, irrelevant, or lacks clear info → return:

json

CopyEdit

{"is\_valid": false, "problem": null, "solution": null}

**EXAMPLE**

Input:  
Title: "My Civic stalls randomly when idling at a red light"  
Body: "2012 Honda Civic, automatic. No error codes. It happens more in hot weather. Dealer couldn’t find anything."  
Top Comment: "Check the idle air control valve (IAC). Cleaning or replacing it fixed mine. Could also be a vacuum leak."

Output:

json

CopyEdit

{

"is\_valid": true,

"problem": "2012 Honda Civic stalls at idle, especially in hot weather. No error codes and no diagnosis from dealer.",

"solution": "Cleaning or replacing the idle air control valve (IAC) usually fixes this. A vacuum leak is another possibility."

}

**📚 Prompt Variant 3: 🧪 \*Highly Structured, Token-Efficient Format for Batching or Quantized Models\***

\*\*Best for large-scale processing or quantized models where prompt size matters.\*\*

```text

# TASK: Extract a car repair problem and its solution from Reddit data.

Context:

This dataset is being cleaned to build a local chatbot that helps drivers and mechanics solve car problems. Only extract valid mechanical issues and actionable solutions. Avoid humor, vague advice, or irrelevant discussion.

Format:

Return a JSON object with:

- "is\_valid" (true/false)

- "problem" (1–3 sentence description)

- "solution" (1–3 sentence fix)

If the post is invalid (not enough context or not repair-related), return:

{"is\_valid": false, "problem": null, "solution": null}

---

Title: {{ title }}

Body: {{ selftext }}

Top Comment: {{ top\_comment }}

---

JSON Output:

{

"is\_valid": ...,

"problem": "...",

"solution": "..."

}

Copy

Edit

✅ Summary of Prompt Types

Prompt Version Focus Best for

🔷 V1 (Original) Balanced, rich instructions General-purpose LLMs (e.g., DeepSeek, Zephyr, LLaMA)

🧠 V2 Chain-of-Thought + mechanic role Reasoning-intensive LLMs

🧪 V3 Compact + efficient format Batched inference, quantized models