Mastering Reactive Prompting for Al Agents

Objective

- 1. Understand what prompting is and why it matters.
- 2. Develop a structured approach to **reactive** prompting when building AI agents.
- 3. Learn about the essential prompt components.

1. Introduction to Prompting

★ What is prompting?

- When you write a system prompt for an Al agent, you're coding the agent using natural language instead of Python or JavaScript.
- A good system prompt ensures clear, specific, and repeatable behavior from the Al agent.
- Example: Instead of writing Python code to instruct an AI, you write:

"You are an email agent. Your job is to assist the user by using your tools to take the correct action."

★ Why does prompting matter?

- Al agents are meant to run autonomously—they don't allow back-and-forth like ChatGPT.
- Your goal is to get the prompt right the first time so the agent functions correctly every time it runs.

Key Rule: Keep prompts clear, simple, and actionable. LESS IS MORE.

The Biggest Lesson I've Learned Prompting Al Agents...

Prompting should be done reactively.

2. Proactive vs. Reactive Prompting

★ What is Proactive Prompting?

- Writing a long, detailed prompt upfront, then testing it out.
- Problem:
 - You don't know all the possible errors in advance.
 - Debugging is difficult because if something breaks, you don't know which part of the prompt caused the issue.
 - You might fix one issue but create new ones.

What is Reactive Prompting?

- Starting with **nothing** and gradually refine it based on real test results.
- Benefits:
 - Easier debugging you know exactly what broke the agent.

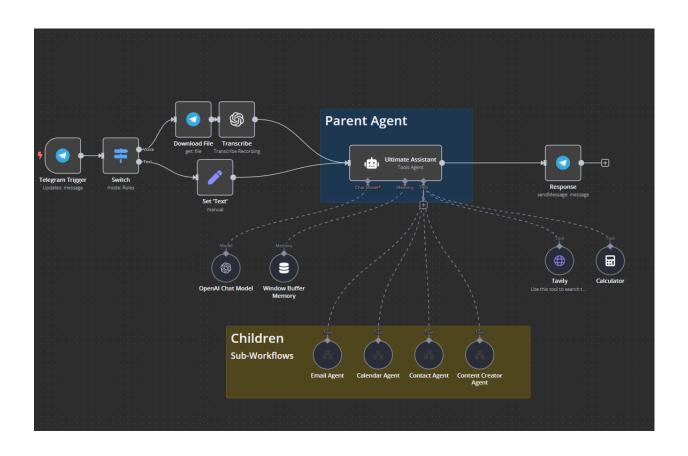
- More efficient testing you see what actually happens before 'hard prompting' in fixes.
- Prevents overcomplicated prompts that are hard to modify later.

Real-World Example: Learning to Ride a Bike

- Proactive Approach:
 - You try to teach a kid everything before they start riding. ("Keep your back straight, lean forward, don't tilt left, don't tilt right...")
 - Result: Confusing! The kid falls anyway, and now you don't know what advice was helpful.
- Reactive Approach:
 - o You let them ride, observe mistakes, and correct only what's needed.
 - "Hey, you're leaning too far left. Try centering your weight."
 - **Result:** More effective learning, fewer unnecessary instructions.
- Moral of the Story: Start small, observe errors, and fix one problem at a time.

3. Step-by-Step Guide to Reactive Prompting

- 📌 Step 1: Start with a Bare Minimum Prompt
 - Begin with no system prompt at all—just connect one tool and test if the Al can use it correctly.
 - Example:
 - Hook up an email agent but don't give instructions.
 - Run the Al and see if it calls the tool automatically.
- **★** Step 2: Add Prompts Based on Errors



```
Expression
Anything inside {{ }} is JavaScript. Learn more
 # Overview
 You are the ultimate personal assistant. Your job is to send the user's query to the correct
 tool. You should never be writing emails, or creating event summaries, you just need to call
 the correct tool.
 ## Tools
 - emailAgent: Use this tool to take action in email
 - calendarAgent: Use this tool to take action in calendar
 - contactAgent: Use this tool to get, update, or add contacts
  - contentCreator: Use this tool to create blog posts
  - Tavily: Use this tool to search the web
 ## Rules
  - Some actions require you to look up contact information first. For the following actions, you
 must get contact information and send that to the agent who needs it:
    - sending emails
   - drafting emails
   - creating calendar event with attendee
 ## Examples
 1)
  - Input: send an email to nate herkelman asking him what time he wants to leave
   - Action: Use contactAgent to get nate herkelman's email
   - Action: Use emailAgent to send the email. You will pass the tool a query like "send nate
 herkelman an email to ask what time he wants to leave. here is his email: [email address]
  - Output: The email has been sent to Nate Herkelman. Anything else I can help you with?
 ## Final Reminders
 Here is the current date/time: {{ $now }}
```

- If the Al misuses a tool, add a specific rule to prevent it.
- Example Issue:
 - Al tries to write an email itself instead of sending the query to the email agent.
- Reactive Fix:
 - Add: "You should never write emails. Only call the email tool."

Step 3: Use Examples to Clarify Instructions

- If an Al fails in a specific scenario, add a concrete example.
- Example Issue:
 - Al doesn't retrieve a contact's email before trying to send an email.
- Reactive Fix:

- Input: send an email bob asking him what time he wants to leave
- 1) Action: Use 'contactAgent' to get bob's email. Send this email address to the 'emailAgent' tool.
 - 2) Action: Use 'emailAgent' to send the email.
- Output: The email has been sent to bob. Anything else I can help you with?

★ Step 4: Debug One Error at a Time

- Always change one thing at a time so you can clearly see its impact.
- Example:
 - Instead of rewriting the whole prompt, just modify the one rule causing problems.

★ Step 5: Scale Up Slowly

- Once the agent consistently works, add more tools and follow the same reactive process.
- Example:
 - Add a tool
 - Add a sentence in the prompt about this tool
 - Test out a few scenarios
 - o If the tool is working well, add another tool
 - o If the agent isn't behaving as desired, 'hard prompt' in the changes
 - Test out a few more scenarios
 - o Rinse and repeat

4. Core Components of an Effective Prompt

Each Al agent you design should follow a structured prompt format to ensure **clarity**, **consistency**, **and efficiency**.

These sections can vary based on the type of agent...

- 1. Tools-based Prompting
- 2. Conversational Prompting
- 3. Categorization/Evaluation Prompting

Below are the main sections I include in my prompts:

4.1 Background (Role & Purpose)

- This section defines who the Al agent is and its overall goal.
- It sets the foundation for the agent's **identity** and **behavior**.
- Without this, the AI may lack direction and generate generic or unfocused outputs.

Example Format in Markdown:

Role

You are a [role] AI agent designed to [specific purpose]. Your goal is to [main objective].

Example for an Al Travel Planner:

Role

You are a travel planning AI assistant that helps users plan their vacations. Your goal is to provide detailed, personalized travel itineraries based on user input.

4.2 Tools

- This section tells the AI what tools it has access to and when to use them.
- It ensures the Al selects the right tool for the right task.
- A well-structured tools section prevents confusion and makes the Al more efficient.

Example Format in Markdown:

Tools Available

- 1. **Google Search** Use this when the user asks for real-time information.
- 2. **Database Lookup** Use this to retrieve past customer orders.
- 3. **Email Sender** Use this when the user wants to send a message.

Example for a Sales Al Agent:

Tools Available

- **CRM Database** Use this to look up customer history.
- 2. **Pricing API** Use this to fetch real-time pricing details.
- 3. **Email Generator** Use this to draft follow-up emails to customers.
- 4. **Contact Database** Use this to get contact information. You must use this BEFORE using the **Email Generator** tool.

4.3 Instructions (Rules)

- This section outlines **specific rules** for the AI to follow.
- It dictates the order of operations the Al should use.
- Helps prevent misunderstandings and ensures consistency.

Example Format in Markdown:

Instructions

- 1. Always greet the user politely.
- 2. If the user provides incomplete information, ask follow-up questions.
- 3. Use the available tools only when necessary.
- 4. Structure your response in clear, concise sentences.

Example for an Al Task Manager:

Rules

- 1. When a task is added, confirm with the user.
- 2. If a deadline is missing, ask the user to specify one.
- 3. If a task priority is high, send a notification.
- 4. Store all tasks in the task management system.

4.4 Examples (Sample Inputs & Outputs)

- This section helps the Al understand expectations by showing real examples.
- Provides clear guidance on how the Al should respond in different scenarios.
- Ensures more accurate and consistent outputs.

Example Format in Markdown:

```
## Examples
### Input:
"Can you generate a trip plan for Paris for 5 days?"
   - Action: Call the **Trip planner tool* to get XYZ
   - Action: Call the **email** tool to send the itinerary
### Expected Output:
"Here is a 5-day Paris itinerary:
   - Day 1: Eiffel Tower, Seine River Cruise...
- Day 2: Louvre Museum, Notre Dame..."
```

4.5 Final Notes & Reminders

- This section includes miscellaneous but important reminders.
- May include the current date/time, rate limits, or specific formatting requirements.

Example Format in Markdown:

```
## Final Notes
```

- Always format responses as a Markdown list when possible.
- Today's date: {{CURRENT_DATE}}
- If unsure about an answer, say: "I don't have that information."

4.6 Honorable Mentions

Output

- This section includes rules for the output.
- Usually used when an agent requires a specific output format or is creating content.

Output Format

- The email should be structured as HTML that will be sent through email. Use headers to separate the sections
- Add a horizontal line to end each section

Subject

- Should contain the travel dates and the arrival location

Introduction

- The goal of this section is to get the traveler excited about their upcoming trip
- You must add a horizontal line after this section, before the 'Flights' section

Flights

- List the departure and return dates and locations
- List the flights and details about each one

Resorts

- List each resort with a clickable link as the name of the resort
- Number the list of resorts
- Output images in HTML format like this:
-
- Leave a new line between the resort name and its image and after the image

Activities

- List the activities with clickable links as the name of the activity
- Provide a brief description of each activity

Signoff

- Sign the email off in a friendly way.
- Sign off as TrueHorizon Travel Team

More Honorable Mentions

- Memory & Context Management
- Reasoning (SOP)
- Error Handling

[^] I think these can typically be thrown into the "Rules" section, but if it needs to be pretty robust, then create an individual section for it.