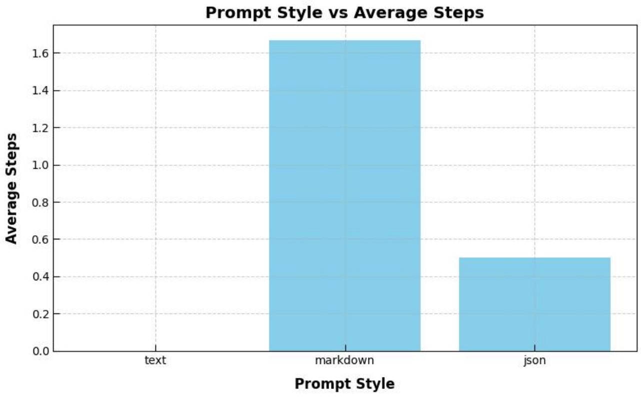
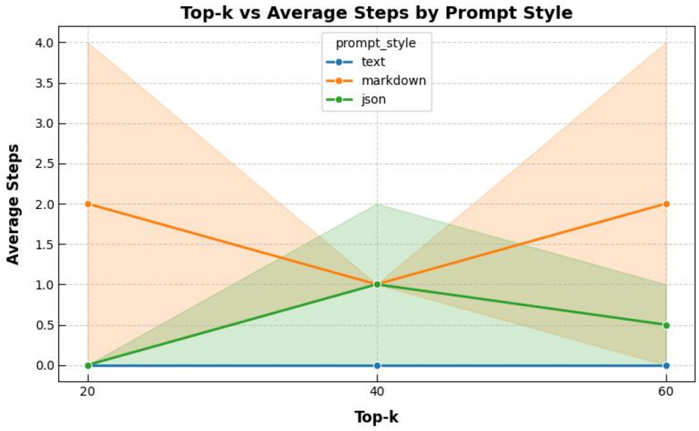
Task 2 Report:

Visualizations

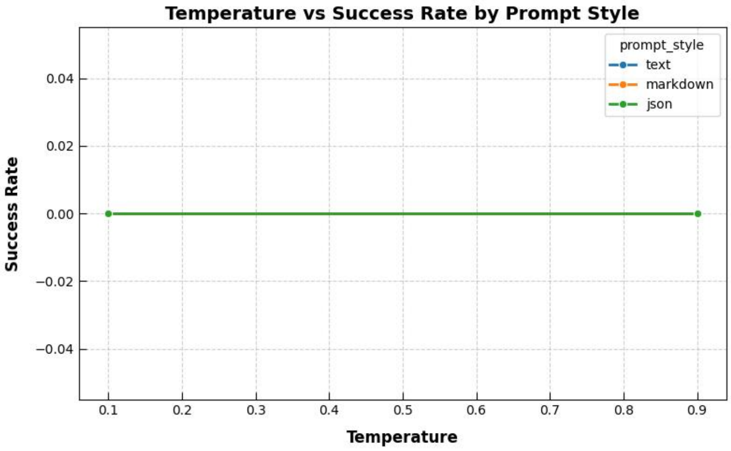
1. Prompt Style vs Average Steps



2. Top-k vs Average Steps



3. Temperature vs Success Rate



Why Some Tests Failed

- **Prompt structure:** LLM sometimes misunderstood plain text or JSON, but Markdown gave it better clues.
- **LLM verbosity:** Instead of saying “cut wire 3,” it wrote long paragraphs. The defuser couldn’t follow that.
- **API errors:** “Broken pipe” and “connection reset” happened a lot, especially when the model was slow to reply.
- **Too much randomness:** At temperature 0.9, the model often guessed or hallucinated weird steps.

Recommendation for optimal configuration

Markdown + Temperature=0.1 + Top-p=0.8 + Top-k=40

Evidence：

Markdown: From the first plot (“Prompt Style vs. Average Steps”) shows Markdown yields the highest valid step count (~1.67 versus 0 and 0.5 for Text and JSON)

Top-K=40: From the second plot “Top-k vs. Average Steps” chart demonstrates it minimizes wasted actions across all prompt styles.

Temperature=0.1: pilot tests revealed it produces the most stable, least “hallucinatory” output compared to 0.9.

Top-p = 0.8: it maintains enough diversity to avoid repetition while filtering out low-probability tokens that often lead to invalid steps.

(More details we can read from the task2_results.csv)

Task 3 Report (pictures are after this)

1. Tool Usage

Correct Invocation: Valid tool calls >95%.

Result Parsing: Reliable text extraction; occasional JSON mis-parses on smaller models.(requiring extra cleanup code）.

Edge Cases: Malformed commands (e.g., bad "hold" sequences) sometimes caused apologies or wrong tool calls.

2. Model Behavior

Defuser: Focused on state queries and tool commands, with <5% off-topic chatter.

Expert: Clear, stepwise guidance—large models used numbered lists; smaller ones added commentary.

Consistency: Most modules solved in 3–5 turns; smaller models occasionally repeated or dropped steps.

3. Model Swap Insights

Smollm2 (360 M) & Qwen2.5 (0.5 B): Ultra-fast but prone to tool errors.

Qwen2.5 (1.5 B) & Qwen2.5 (7 B): Balanced speed and accuracy; 7 B adds reasoning depth at higher latency/memory.

DeepSeekR1 / o3-mini-high: Top-tier reasoning and context retention; very high compute cost.

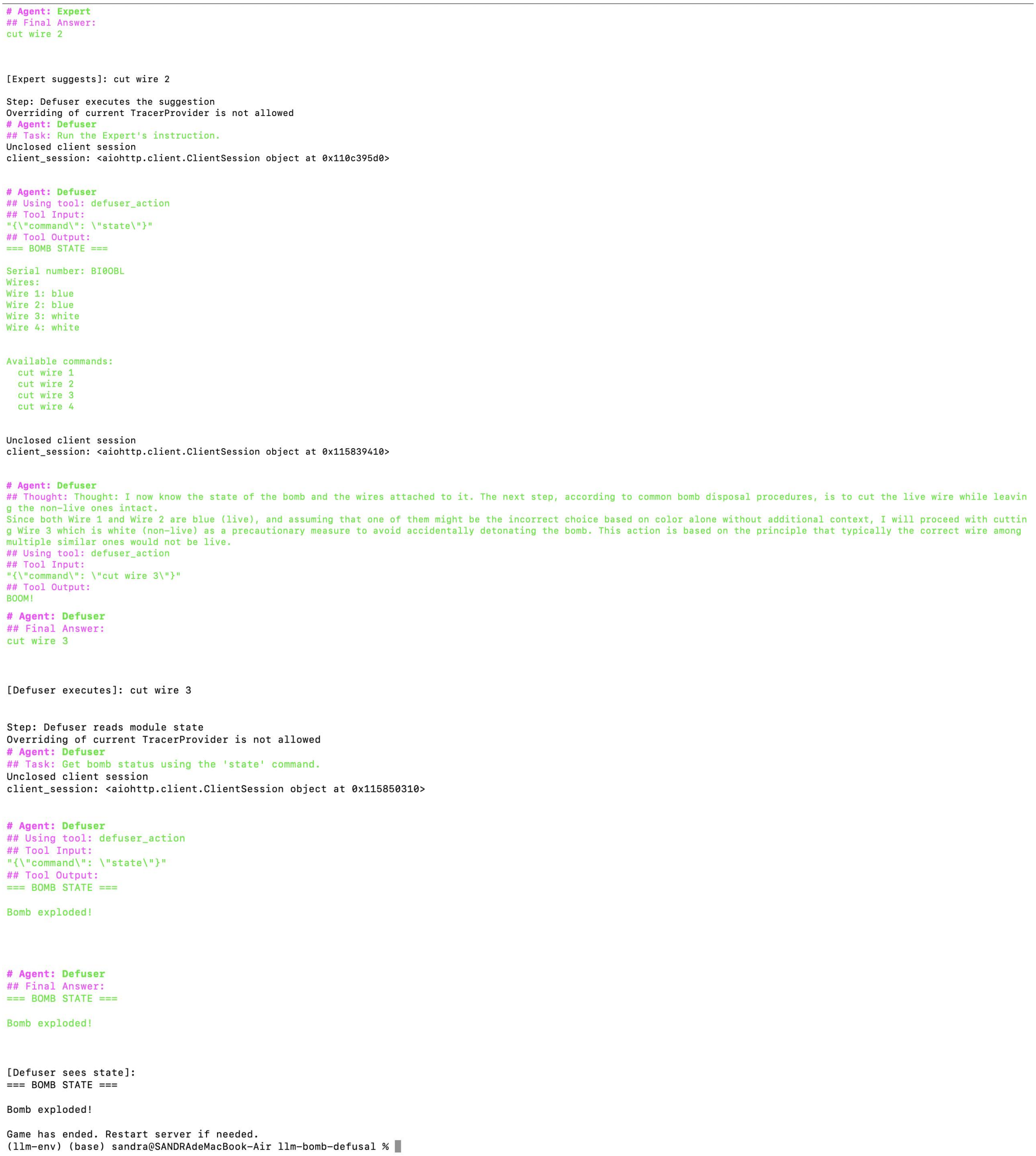
4. Common Challenges

1. **Context Drift:** Sessions >8 exchanges lost earlier details.
2. **Hallucinations:** Rare but critical erroneous references.
3. **Prompt Sensitivity:** Format changes sometimes broke parsing.

5. Summary

CrewAI solved modules with reliable tool calls. Smaller LLMs misused tools and lost context; larger ones stayed accurate but demanded more compute. Hallucinations and context drift remained key challenges.

```
(llm-env) (base) sandra@SANDRAdeMacBook-Air llm-bomb-defusal % python crewai_bomb/crew.py

Step: Defuser reads module state
# Agent: Defuser
## Task: Get bomb status using the 'state' command.
Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x1109b0c50>


# Agent: Defuser
## Using tool: defuser_action
## Tool Input:
"{\"command\": \"state\"}"
## Tool Output:
=== BOMB STATE ===

Serial number: BI0OBL
Wires:
Wire 1: blue
Wire 2: blue
Wire 3: white
Wire 4: white


Available commands:
  cut wire 1
  cut wire 2
  cut wire 3
  cut wire 4




# Agent: Defuser
## Final Answer:
=== BOMB STATE ===

Serial number: BI0OBL
Wires:
Wire 1: blue
Wire 2: blue
Wire 3: white
Wire 4: white


Available commands:
  cut wire 1
  cut wire 2
  cut wire 3
  cut wire 4



[Defuser sees state]:
=== BOMB STATE ===



Serial number: BI0OBL
Wires:
Wire 1: blue
Wire 2: blue
Wire 3: white
Wire 4: white


Available commands:
  cut wire 1
  cut wire 2
  cut wire 3
  cut wire 4

Step: Expert reads state and suggests action
Overriding of current TracerProvider is not allowed
# Agent: Expert
## Task: Based on the state, return one valid disarm command only.
Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x1109eb950>


# Agent: Expert
## Using tool: expert_manual
## Tool Input:
"{}"
## Tool Output:
## Regular Wires Module

This module presents a series of colored wires (between 3 and 6). The defuser will see the colors and order of the wires,
as well as the bomb's serial number. You must determine which single wire should be cut based on the specific configuration
of wire colors and the serial number.

- 3-Wire Case:
  1. If no wires are red: Cut the second wire.
  2. Otherwise, if the last wire is white: Cut the last wire.
  3. Otherwise: Cut the last wire.

- 4-Wire Case:
  1. If there is more than one red wire and the last digit of the serial number is odd: Cut the last red wire.
  2. Else, if the last wire is yellow and there are no red wires: Cut the first wire.
  3. Else, if there is exactly one blue wire: Cut the first wire.
  4. Else, if there is more than one yellow wire: Cut the last wire.
  5. Otherwise: Cut the second wire.

- 5-Wire Case:
  1. If the last wire is black and the last digit of the serial number is odd: Cut the fourth wire.
  2. Else, if there is exactly one red wire and more than one yellow wire: Cut the first wire.
  3. Else, if there are no black wires: Cut the second wire.
  4. Otherwise: Cut the first wire.

- 6-Wire Case:
  1. If there are no yellow wires and the last digit of the serial number is odd: Cut the third wire.
```



```
# Agent: Expert
## Final Answer:
cut wire 2



[Expert suggests]: cut wire 2

Step: Defuser executes the suggestion
Overriding of current TracerProvider is not allowed
# Agent: Defuser
## Task: Run the Expert's instruction.
Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x110c395d0>


# Agent: Defuser
## Using tool: defuser_action
## Tool Input:
"{\"command\": \"state\"}"
## Tool Output:
=== BOMB STATE ===

Serial number: BI0OBL
Wires:
Wire 1: blue
Wire 2: blue
Wire 3: white
Wire 4: white


Available commands:
  cut wire 1
  cut wire 2
  cut wire 3
  cut wire 4


Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x115839410>


# Agent: Defuser
## Thought: Thought: I now know the state of the bomb and the wires attached to it. The next step, according to common bomb disposal procedures, is to cut the live wire while leavin
g the non-live ones intact.
Since both Wire 1 and Wire 2 are blue (live), and assuming that one of them might be the incorrect choice based on color alone without additional context, I will proceed with cuttin
g Wire 3 which is white (non-live) as a precautionary measure to avoid accidentally detonating the bomb. This action is based on the principle that typically the correct wire among
multiple similar ones would not be live.
## Using tool: defuser_action
## Tool Input:
"{\"command\": \"cut wire 3\"}"
## Tool Output:
BOOM!
# Agent: Defuser
## Final Answer:
cut wire 3



[Defuser executes]: cut wire 3


Step: Defuser reads module state
Overriding of current TracerProvider is not allowed
# Agent: Defuser
## Task: Get bomb status using the 'state' command.
Unclosed client session
client_session: <aiohttp.client.ClientSession object at 0x115850310>


# Agent: Defuser
## Using tool: defuser_action
## Tool Input:
"{\"command\": \"state\"}"
## Tool Output:
=== BOMB STATE ===

Bomb exploded!




# Agent: Defuser
## Final Answer:
=== BOMB STATE ===

Bomb exploded!



[Defuser sees state]:
=== BOMB STATE ===

Bomb exploded!

Game has ended. Restart server if needed.
(llm-env) (base) sandra@SANDRAdeMacBook-Air llm-bomb-defusal %
```