

## **Analysis Memo:**

Overall, the goal of Phase 1 was to get a baseline understanding of how different LLMs handle core research tasks and how prompt design affects grounding, citation behavior, and reliability. Specifically, in the domain of Martian habitability and biosignatures. Two models were evaluated, Gemini Think and Copilot Smart, on two tasks using a baseline prompt (Prompt A) and a structured prompt with guardrails (Prompt B).

A clear pattern across all experiments was that prompt design mattered more than model choice. Gemini Think and Copilot Smart generally produced similar-quality outputs when given the same prompt. However, Prompt B consistently produced more reliable, structured, and auditable outputs than Prompt A. Initially, Prompt A often generated fluent and reasonable summaries or claims, but rarely included citations or explicit quotes between claims and evidence. In contrast, Prompt B almost always produced outputs that followed the requested structure and included citations for each field or claim.

However, over time, both models began to mirror the formatting and structure I had been using in previous interactions, even though each prompt was started in a new chat. This likely reflects interface-level priming or conversational carryover rather than true generalization. As a result, some later Prompt A runs appeared stronger than earlier ones and occasionally resembled Prompt B outputs. This makes manual chat-based evaluation less reliable and highlights the importance of running models in isolated, script-based environments in later phases.

For the paper triage task, Prompt A produced readable summaries that were useful for getting a quick sense of a paper, but these summaries were not research-grade because claims could not be traced back to evidence. Prompt B summaries were sometimes more rigid and less polished, but they consistently included citations and therefore supported verification. This revealed a tradeoff between readability and auditability, with auditability being more important for a research system.

For claim–evidence extraction, the difference between Prompt A and Prompt B was less pronounced for Copilot Smart than for Gemini Think. Copilot Smart often produced reasonable claims with supporting text and sometimes included source-level or in-text citations even under Prompt A, although these were not consistently placed or formatted in the required table structure. Gemini Think, on the other hand, rarely included citations under Prompt A. Prompt B reliably enforced the Claim | Quote | Citation schema for both models, producing outputs that could function as structured research artifacts.

Several failure modes were observed. First, models defaulted to producing confident statements unless explicitly constrained. Second, none of the models triggered the fallback instruction to output fewer than five claims and state “Insufficient evidence in provided excerpt.” Even when such behavior was requested, models always attempted to generate five claims by splitting or rephrasing ideas. This suggests that LLMs prioritize satisfying output format over assessing evidence sufficiency. Third, uncertainty was inconsistently expressed; models sometimes used hedging language for modeling results but rarely attached uncertainty to broader habitability claims.

These findings directly inform Phase 2 design choices. Phase 2 will rely on retrieval-grounded citations rather than self-reported citations from the model. All answers will be generated from retrieved chunks

with explicit source and chunk identifiers. Prompts will continue to enforce structured schemas similar to Prompt B. Additionally, evidence-gating logic will be introduced: if retrieval returns insufficient evidence, the system should refuse to answer or explicitly state that evidence is missing. Finally, Phase 2 evaluations will be run via scripts with logging to avoid conversational priming and ensure reproducibility.

Overall, Phase 1 shows that large language models can support research-style workflows, but only when heavily scaffolded. Prompting alone improves behavior, but true reliability will require retrieval, structured outputs, and system-level constraints in later phases.