MARRVEL-MCP REVISION

Hyun-Hwan Jeong

Liu Lab Meeting

February 4, 2026

Revision Summary

Reviewer #1 *6 comments*

- Positive: timely and important topic
- Wants stronger positioning vs. existing AI agents
- Requests more evaluation details and baselines
- Terminology and title wording concerns

Reviewer #2 *6 comments*

- Positive: solid engineering, useful MCP tools
- Wants more honest framing of contributions
- Requests deeper evaluation and tool analysis
- Concerns about gold standard reliability

Reviewer #1: Overall Comments

This manuscript describes MARRVEL-MCP, an AI-agent system that automatically navigates appropriate biomedical resources to answer rare disease questions. This is a ***very timely and important topic*** *for Mendelian disease discovery in the era of AI. The reviewer has several suggestions for improving the manuscript.*

6 specific comments:

1. Related work on biomedical AI agents
2. Clarification of “context engineering”
3. Benchmark dataset and evaluation details
4. Comparison with commercial AI tools (GPT, Gemini)
5. Title wording clarification
6. Usability improvement evidence

R1.1: Related Work on Biomedical AI Agents

Reviewer comment:

The authors should consider including more relevant work on biomedical AI agents (e.g., BioMNI). It would also be helpful to compare MARRVEL-MCP with existing biomedical AI agents in terms of architecture, performance, speed, etc., to better highlight the uniqueness of the proposed system for rare diseases.

Response:

We thank the reviewer for this suggestion. We have expanded the Background section to include recent biomedical AI agent systems—BioMNI, RDguru, STELLA, and KGAREVION—and added a systematic comparison table (Table 1) contrasting architecture, model requirements, data granularity, and design characteristics. We also clarified MARRVEL-MCP's unique positioning as a system that enables lightweight, off-the-shelf models (3B–20B) to achieve expert-level genomic workflows through context engineering rather than frontier-scale compute.

R1.1: Added Comparison Table

Table 1: Comparison of MARRVEL-MCP with recent biomedical AI agent frameworks.

System	Primary Focus	Model Architecture	Data Granularity	Design Characteristics
BioMNI	Multimodal reasoning & dynamic workflow	Agentic module (BioMNI-A1) with frontier models (Claude, GPT-4)	Broad biomedical concepts & literature	Sophisticated reasoning; requires high-resource infrastructure
STELLA	Self-improving biomedical reasoning	Dynamic tool discovery with BioMNI-A1 backbone	Broad biomedical resources	Adaptive self-evolution; frontier model dependency
KGAREVION	Knowledge-intensive QA via graph verification	LLM verification against static KGs (PrimeKG)	Gene-disease associations from ontologies	Graph-based verification; gene/disease level
RDguru	Rare disease clinical consultation	Knowledge-based conversational interface	Disease-symptom associations	Clinical dialogue support
MARRVEL-MCP	**Variant interpretation & gene analysis**	**Context-engineered tools with 3B–20B models (no fine-tuning)**	**Gene/variant-level granularity**	**Lightweight models via structured DB access; variant-level precision**

R1.1: Revised Manuscript Text

Added to Background:

More recently, specialized biomedical AI agents have emerged as a transformative paradigm for clinical applications, moving beyond passive question-answering to active problem-solving through autonomous tool use and multi-step reasoning [Gao 2024]. These systems include BioMNI for multimodal reasoning through literature-mining dynamic workflow composition [Huang 2025], RDguru for rare disease diagnosis through knowledge-based question answering and clinical consultation support [Yang 2025], STELLA for self-improving biomedical reasoning through dynamic tool discovery and evolving reasoning templates [Jin 2025], and KGAREVION for knowledge-intensive medical question answering through knowledge graph verification of LLM-generated information [Su 2024].

*We systematically compare MARRVEL-MCP with these frameworks (see Table X) and illustrate how our architecture enables **lightweight, off-the-shelf models** to match the utility of high-resource generalist agents through specialized context engineering. While systems like BioMNI and STELLA demonstrate impressive reasoning, their agentic modules (e.g., BioMNI-A1) depend on computationally intensive frontier models (e.g., Claude, GPT-4) to orchestrate complex tasks, limiting their deployability in local or low-resource environments. Conversely, approaches like KGAREVION rely on static knowledge graphs (PrimeKG) [Chandak 2023] that lack the variant-level granularity—e.g., specific genomic coordinates and allele frequencies—required for pathogenicity diagnosis.*

*MARRVEL-MCP breaks this trade-off between model accessibility and analytical depth. By defining a rigorous "context engineering" environment, we establish a new paradigm: **enabling lightweight, off-the-shelf models (3B–20B) to autonomously execute expert-level genomic workflows without the need for massive scale or specialized fine-tuning**. This approach effectively offloads the requirement for "intelligence" from the model parameters to the tool environment, allowing small local agents to perform with the precision of frontier-scale systems.*

R1.2: Clarification of "Context Engineering"

Reviewer comment:

> *The term context engineering has been used to describe the next generation of RAG systems. Its definition typically extends beyond selecting appropriate input resources, which appears to be the main definition used in this study. Please ensure that the usage in the manuscript is precise and consistent with existing terminology.*

Response:

We thank the reviewer for raising this important point. We have revised the manuscript across the Abstract, Background, and Discussion sections to adopt the broader, formally recognized definition of context engineering [Mei 2025]. We now explicitly distinguish context engineering from conventional RAG and from general-purpose agent frameworks, and clarify that our work focuses specifically on tool-mediated autonomous analysis as one instantiation within the broader context engineering taxonomy. The term "tool-augmented context engineering" is now used consistently throughout.

R1.2: Revised Abstract

Revised in Abstract:

*This work demonstrates the impact of **tool-augmented context engineering**—the deliberate design of domain-aware tool environments and structured information scaffolding through executable function interfaces—in reshaping the role of model scale in genomics. MARRVEL-MCP equips LLMs with 39 tools spanning gene and variant utilities, pathogenicity databases, phenotype resources, expression atlases, ortholog data, and literature APIs. Without hard-coded pipelines, LLMs autonomously infer workflows, performing named-entity recognition, identifier normalization, and multi-database synthesis from narrative queries.*

R1.2: Revised Background

Added to Background:

Context-engineering, as recently formalized, encompasses the systematic optimization of information payloads for LLMs through the structured assembly of multiple context components, including instructions, knowledge, tools, memory, and system state [Mei 2025]. These multiple context components fundamentally distinguish context engineering from conventional retrieval-augmented generation (RAG). While conventional RAG retrieves relevant text passages to augment prompts, it remains constrained by static workflows, lacks executable tools and structured processes, and provides insufficient adaptability for multi-step reasoning and complex task management [Singh 2025].

Context-engineering also differs from general-purpose agent frameworks such as AutoGPT and ReAct, which provide planning scaffolds but lack the domain-specific tools and constraints necessary for biological reasoning. A general agent might generate a plan like "Step 1: Search for variant pathogenicity. Step 2: Summarize findings"—but without access to ClinVar's API and knowledge of HGVS notation, it cannot execute this plan reliably.

Applying these principles, our work focuses specifically on ***tool-mediated autonomous analysis****: we provide LLMs with executable function interfaces to curated databases, enabling them to discover and select appropriate analytical pathways rather than following predefined workflows. By designing tool interfaces that communicate not just what they compute but when and why they should be used in biological investigation, we enable the LLM to autonomously navigate complex analytical decisions analogous to those made by experienced computational biologists.*

R1.2: Revised Discussion (1/2)

Added to Discussion:

MARRVEL-MCP represents a specific instantiation of context engineering focused on tool-mediated database access. Recent taxonomies of context engineering [Mei 2025] identify several complementary approaches to optimizing LLM information payloads: (1) RAG systems that retrieve and re-rank text documents, (2) memory systems that maintain conversation state, (3) tool-integrated reasoning that provides executable functions, and (4) multi-agent orchestration for complex workflows. Our work focuses on "tool-integrated reasoning"—demonstrating that context quality can compensate for model scale, regardless of whether the context originates from retrieved documents or structured database queries.

MARRVEL-MCP demonstrates that structured tool environments can reshape the relationship between model scale and genomic task performance. A 20B-parameter model with MARRVEL-MCP achieved 95% accuracy compared to 33% without tools—matching systems 5–10$\times$ larger and approaching the performance of state-of-the-art proprietary models. This pattern held consistently: lightweight models with appropriate context outperformed much larger systems operating without tool access.

This effect stems from context engineering—encoding domain constraints, nomenclature standards, coordinate systems, and provenance metadata directly into the tool environment. The rs193922679 example crystallizes this principle: Haiku with MARRVEL-MCP delivered structured dbNSFP predictions formatted for clinical review by autonomously inferring the correct workflow (rsID conversion, coordinate-based query, tabular summary), while a much larger model without tools provided only narrative web search results.

R1.2: Revised Discussion (2/2)

Discussion (continued):

The key insight generalizes: the systematic design of information structure and access patterns enables smaller models to achieve performance comparable to frontier systems on domain-specific tasks. Future genomic assistants might benefit from combining these approaches: using RAG for literature synthesis and clinical note analysis, while employing tool-based systems like MARRVEL-MCP for structured variant annotation and database queries. Such hybrid architectures would leverage the complementary strengths of different context engineering strategies—the semantic flexibility of retrieval with the computational precision of structured tools.

Revised Conclusion:

This work demonstrates that tool-augmented context engineering—the systematic design of domain-specific function interfaces and structured information flows—can compensate for limited model capacity in specialized domains. By optimizing how genomic information is accessed and formatted rather than relying solely on model scale, MARRVEL-MCP enables lightweight models to achieve accuracy comparable to frontier systems on variant interpretation tasks. This finding generalizes beyond genomics: context engineering, whether through document retrieval, tool integration, or hybrid approaches, represents a fundamental strategy for building reliable, efficient domain-specific AI systems. Future work should explore integration of multiple context engineering modalities to leverage their complementary strengths.

R1.3: Benchmark Dataset and Evaluation Details

Reviewer comment:

Please provide additional details about the benchmark dataset and evaluation pipeline. What criteria were used to define "answerable" questions? Were answers manually generated by experts? When Claude Sonnet was used as an external judge, did it compare with expert-generated reference answers, or rely on its own internal knowledge?

Response:

We thank the reviewer for this question. We have added a new Implementation subsection detailing the benchmark construction and evaluation pipeline. The benchmark of 100 questions was manually constructed and categorized into baseline, single-tool, and multi-tool queries. Expected answers were curated by team members with expertise in molecular genetics. Claude Sonnet 4 was used as an automated judge that directly compared LLM responses against the human-curated expected answers—not its own internal knowledge. For time-sensitive questions, automated judgments were supplemented with manual review. The evaluation code is publicly available.

R1.3: Revised Implementation – Benchmark Dataset

Added to Implementation:

We manually constructed a benchmark dataset of 100 questions designed to evaluate variant interpretation capabilities answerable using MARRVEL data sources. The benchmark encompassed diverse query types encountered in clinical genetics, from basic gene function lookups to complex multi-source queries requiring integration of variant databases, model organism phenotypes, protein interactions, and recent publications. Questions were categorized into two types based on the complexity of tool calling usage: (1) baseline queries answerable without MARRVEL-MCP using pre-trained knowledge, (2) single-tool queries requiring a single MARRVEL-MCP tool call, and (3) multi-tool queries requiring information synthesis across multiple sequential tool calls and data sources.

For each question, expected answers were manually curated by members of our team with expertise in molecular genetics and variant interpretation. These reference answers were created by systematically querying MARRVEL directly and synthesizing the returned information into concise, accurate responses. Each reference answer included key details that served as critical criteria for evaluating correctness. For time-sensitive questions—such as those involving recent PubMed publications—reference answers cannot be predetermined, so we manually reviewed each LLM response against the current state of the data sources at the time of evaluation.

R1.3: Revised Implementation – Evaluation Pipeline

Added to Implementation:

The evaluation pipeline consisted of three main stages: response generation, automated judging with manual review, and statistical analysis.

Response Generation: *Each of the nine LLMs was tested under two conditions: (1) baseline without MARRVEL-MCP access, and (2) tool-enabled with full access to all 39 tools. All API calls were logged to ensure reproducibility and to analyze tool usage patterns.*

Automated Judging with Manual Review: *We employed Claude Sonnet 4 as an automated judge. For each test case, the judge was presented with: (1) the original question, (2) the human-curated expected answer, and (3) the LLM-generated response.* ***Crucially, the judge directly compared the LLM response with the human-curated expected answer rather than relying on its own internal knowledge.*** *The judging prompt instructed Claude Sonnet 4 to verify that all key facts were present, check for factual contradictions, allow different phrasings, mark hallucinated information as incorrect, and account for multi-step reasoning. For dynamic data sources, automated judgments were supplemented with manual review.*

Pass Rate Calculation: *For each model and condition, we calculated the pass rate as the proportion of questions where the LLM response was judged to agree with the human-curated expected answer.*

R1.3: Revised Result Section

Added to Results:

Equipping lightweight models with MARRVEL-MCP substantially improved their accuracy on variant interpretation tasks, enabling smaller models to match or exceed the baseline performance of much larger systems. We evaluated this effect across nine LLMs ranging from 3B to 235B parameters using a benchmark of 45 expert-curated questions answerable using MARRVEL data. Each model was tested in two conditions: without MARRVEL-MCP access (baseline) and with full access to all 39 tools. Responses were evaluated by comparing them to an expert-generated truth set of answers. Claude Sonnet 4 was used to judge if the LLM response agreed with the expert response, with pass rate defined as the proportion of questions judged correct.

R1.4: Comparison with Commercial AI Tools

Reviewer comment:

The authors should include the performance of state-of-the-art commercial AI tools such as GPT and Gemini on the benchmark dataset. If these systems already achieve strong performance, please justify the added value of MARRVEL-MCP (e.g., smaller models, lower cost, domain customization, transparency, or reproducibility).

Action items:

- Benchmark GPT-4o / GPT-o1 and Gemini on evaluation set
- Report performance comparison table
- Justify added value: domain tools, cost, transparency, reproducibility

R1.5: Title Wording Clarification

Reviewer comment:

> *Is MARRVEL-MCP primarily a question-answering system, or does it function as a chatbot that supports multi-turn conversations? I suggest replacing "NATURAL-LANGUAGE QUERY-TO-RESPONSE INTERFACE" in the title with "Question-Answering System" or "Chatbot" to improve clarity and precision.*

Response:

We appreciate this suggestion. We have revised the title to: **"MARRVEL-MCP: An Agentic Interface for Mendelian Disease Discovery via Tool-Augmented Context Engineering."** We chose "Agentic Interface" rather than "Question-Answering System" or "Chatbot" because MARRVEL-MCP is not a simple conversational bot—it is a system where the LLM autonomously selects and orchestrates 39 domain-specific tools to perform multi-step genomic analysis. The term "interface" emphasizes its role as a bridge between the user and structured biological databases, while "agentic" captures the autonomous tool-use capability that distinguishes it from passive QA systems. This framing highlights both the interaction design and the underlying tool orchestration that define the system's contribution.

R1.6: Usability Improvement Evidence

Reviewer comment:

The authors have noted that user-friendliness is one of the motivations for developing MARRVEL-MCP. However, is there any evidence showing that MARRVEL-MCP provides outputs that are more beneficial to users? If there has not been a formal user study, the authors should consider addressing this point briefly in the discussion section.

Action items:

- Acknowledge lack of formal user study (if applicable)
- Add discussion paragraph on usability benefits
- Consider citing indirect evidence or anecdotal feedback
- Mention user study as future work

Reviewer #2: Overall Comments

> *I think this paper is on the right track, but right now it feels more like a **solid engineering effort** than a big conceptual leap. The MCP tools are genuinely useful, but the manuscript could be more honest about what is new versus what is just well-integrated. The MCP layers the authors built are probably the **most valuable part** of the work. They make it much easier for LLMs to interact with biological databases like dbNSFP, which is great for the community. The tooling infrastructure is the most compelling contribution, but the **system-level claims need stronger justification** and clearer evidence.*

6 specific comments:

1. Context engineering already in most agent systems
2. “Hard-coded” vs. non-hard-coded clarification
3. Evaluation is thin; hand-picked examples
4. Which MCP tools help the most?
5. Error types count mismatch (Sec. 3.3)
6. Gold standard reliability concern

R2.1: Context Engineering in Agent Systems

Reviewer comment:

In most agent systems, context engineering is already part of the architecture, so the comparison in the intro is confusing.

Response:

We thank the reviewer for this observation. We have substantially revised the Background to clarify what distinguishes our use of context engineering from standard agent-system context handling. We now explicitly distinguish context engineering from (1) conventional RAG, which lacks executable tools and structured processes, and (2) general-purpose agent frameworks such as AutoGPT and ReAct, which provide planning scaffolds but lack domain-specific tools and constraints. We define our contribution as **tool-mediated autonomous analysis**—providing LLMs with executable function interfaces that communicate not just *what* they compute but *when* and *why* they should be used in biological investigation. This reframes our novelty: not context engineering per se, but domain-specific context engineering that embeds biological reasoning constraints into the tool environment.

R2.1: Revised Background (1/2)

Added to Background:

Context-engineering, as recently formalized, encompasses the systematic optimization of information payloads for LLMs through the structured assembly of multiple context components, including instructions, knowledge, tools, memory, and system state [Mei 2025]. These multiple context components fundamentally distinguish context engineering from conventional retrieval-augmented generation (RAG). While conventional RAG retrieves relevant text passages to augment prompts, it remains constrained by static workflows, lacks executable tools and structured processes, and provides insufficient adaptability for multi-step reasoning and complex task management [Singh 2025].

Context-engineering also differs from general-purpose agent frameworks such as AutoGPT and ReAct, which provide planning scaffolds but lack the domain-specific tools and constraints necessary for biological reasoning. A general agent might generate a plan like "Step 1: Search for variant pathogenicity. Step 2: Summarize findings"—but without access to ClinVar's API and knowledge of HGVS notation, it cannot execute this plan reliably.

R2.1: Revised Background (2/2)

Background (continued):

*Applying these principles, our work focuses specifically on **tool-mediated autonomous analysis**: we provide LLMs with executable function interfaces to curated databases, enabling them to discover and select appropriate analytical pathways rather than following predefined workflows. This approach exemplifies how context-engineering principles—optimizing which information is available and how it is structured—can be instantiated through domain-specific tooling that exposes biological purpose alongside computational function. By designing tool interfaces that communicate not just what they compute but when and why they should be used in biological investigation, we enable the LLM to autonomously navigate complex analytical decisions analogous to those made by experienced computational biologists. This creates a system in which domain knowledge is embedded in both the tool environment and the contextual framework that guides tool selection, enabling reliable computational workflows while preserving adaptability in biological problem-solving.*

R2.2: Hard-Coded vs. Non-Hard-Coded

Reviewer comment:

Authors emphasized not "hard-coded"; I partially agree. But given many coding is implemented in the MCP layer (providing interfaces), I won't say it is completely non hard-coded. The improvement of MCP over the baseline is because of the "hard-coding" that exposes the abstract layer usable by LLM agents. A better way to describe: MCP is certainly hard-coded, but the workflow is not.

Action items:

- Adopt nuanced framing: MCP tools are hard-coded interfaces, but the **workflow/reasoning** is not hard-coded
- Revise manuscript language accordingly

R2.3: Evaluation Concerns

Reviewer comment:

The current evaluation is a bit thin. The examples in Section 3.1 are hand-picked and likely biased. For the 45 questions, it's not clear which ones actually need MCP tools and which ones don't. I'd really like to see cases when questions do not need MCP tools, how it performed comparing to the baseline.

Action items:

- Bump benchmark from 45 to 100 questions
- Finalizing the benchmark revision
- Check the list of questionnaires that would be answered by LLM or not

R2.4: Tool Usage Analysis

Reviewer comment:

It would be helpful to know which MCP tools actually help the most? How many tool calls are needed per question? Whether performance drops as tool usage increases?

Action items:

- Report per-tool contribution to answer quality
- Analyze distribution of tool calls per question
- Investigate performance vs. number of tool calls
- Add tool usage breakdown figure or table

R2.5: Error Types Count Mismatch

Reviewer comment:

> *There's a small error where the paper says there are three error types but only lists two. (Section 3.3: "We categorized errors into three types based on their underlying cause.")*

Response:

We thank the reviewer for catching this error. We have corrected the count in Section 3.3 to match the number of error types actually listed. We have also conducted a more careful proofreading pass across the entire manuscript to ensure consistency between numerical references and their corresponding content throughout.

R2.6: Gold Standard Reliability

Reviewer comment:

As LLMs become stronger, the reliability of purely human-curated "gold standard" answers becomes less clear. It would be helpful to know whether the authors manually reviewed LLM-generated "error" answers to confirm they are truly incorrect, rather than alternative valid answers.

Action items:

- Manually review LLM answers marked as "errors"
- Report how many "errors" were actually valid alternative answers
- Discuss limitations of human-curated gold standards
- Consider inter-annotator agreement or adjudication process