GARFIELD: GRAPH AUGMENTED REASONING FRAMEWORK FOR INTEGRATED EXPERIMENTAL LAB DISCOVERY

Anonymous authors
Paper under double-blind review

ABSTRACT

Experimental protocols form the knowledge backbone of laboratory experiments and automation. They contain information on experimental steps, actions, materials, instruments, parameters, and execution flow, while also sharing recurring concepts across protocols. This structure and cross-protocol knowledge remain largely implicit in natural language. To address this, we introduce GARFIELD (Graph Augmented Reasoning Framework for Integrated Experimental Lab Discovery), a framework that provides experimental protocols as structured context through our proposed multi-view experimental protocol graph representation. This graph representation captures four views: hierarchical, execution, action context, and cross-protocol alignment. The hierarchical view organizes protocols into processes, steps, and actions; the execution view captures procedural flow; the action context view connects actions with their experimental objects and parameters; and the cross-protocol alignment view links equivalent actions, materials, instruments, products (outputs), and biological samples across protocols. During inference, our graph-aware GARFIELD agent dynamically interacts with the multi-view experimental protocol graph representation with a dedicated set of tools to adaptively retrieve the necessary context based on the query. Experiments show that GARFIELD outperforms language-model and retrieval baselines in retrieving experimental details (+11.5% accuracy), reasoning over questions on structural relationships (+0.326 Token-F1), detecting protocol errors (+11.6% accuracy), and resolving procedural order (+14.4% sequence EM). We also evaluate GARFIELD on protocol design, where access to its structured context improved overlap with gold protocols for experimental operations and execution flow by 27.0% and 13.6%, respectively. These results show that GARFIELD can complement large language models by providing structured experimental context for both protocol knowledge retrieval and design.

1 INTRODUCTION

Laboratory autonomy requires connecting scientific reasoning with reliable execution of experiments. In biological sciences, lab protocols record procedural knowledge needed to interpret, reproduce, and adapt experimental work. Their utility depends on relationships and internal structure; an action belongs to a larger process, acts on particular objects, and is constrained by earlier operations and local conditions. Scientific agents increasingly assist with hypothesis generation, data analysis, and experimental planning (Swanson et al., 2025; Huang et al., 2026; Gottweis et al., 2026). Making documented procedural knowledge accessible is one component of this broader effort toward human-led laboratory autonomy (Cheng et al., 2026), as protocols record the outcome of a successful iterative procedure that included refinement, troubleshooting, and adaptation to a scientific question.

Protocol understanding is a representation and retrieval challenge because connections between objects and actions are frequently implicit across experimental steps. Accurately tracing a substance throughout a procedure requires a scientific agent to follow the flow of the experiment and correctly link the final actions to the initial materials. For instance, in a Golgi-isolation protocol, the target material is initially captured on magnetic beads while the first "supernatant" (the flow-through) is set aside; the beads are later lysed to produce a final supernatant, which is then renamed and measured as

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

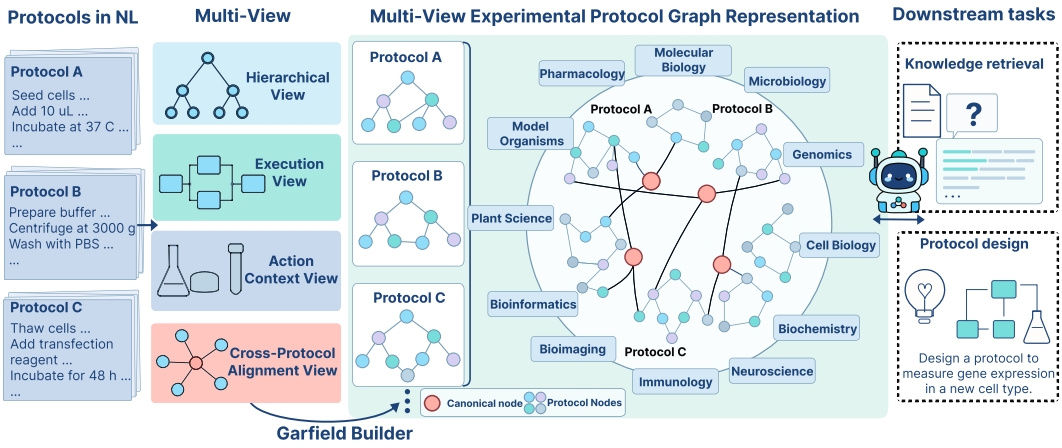

Figure 1: *Abstract representation of* GARFIELD *framework.* During ingestion, natural-language experimental protocols are structured into a multi-view experimental protocol graph representation comprising four views: Hierarchical, Execution, Action Context, and Cross-Protocol Alignment. During inference, a Graph-Aware GARFIELD Agent uses dedicated graph tools to traverse and retrieve structured context, supporting downstream tasks such as knowledge retrieval and protocol design.

the “solubilised Golgi.” Now, a question like, "Which specific prior steps generated the “solubilised Golgi”?" requires resolving the ambiguous term "supernatant" across multiple collection and discard steps. While human experts naturally track these shifting relationships, AI agents struggle with this understanding (Sivakumar et al., 2026). Cross-protocol comparisons introduce further complexity, as recurring concepts must be linked without losing their local experimental intent. A “supernatant” for instance, may be retained if it carries the target biological material, or discarded if the target resides in the pellet. Normalizing these steps requires establishing a *shared identity* for common concepts while maintaining their *local experimental context*.

Existing protocol representations capture dependencies and hierarchy within procedures (Shi et al., 2024b; 2025), while graph-based retrieval supports associative search and traversal across documents (Gutiérrez et al., 2025; Ma et al., 2025). Here, we combine these approaches in a Graph Augmented Reasoning Framework for Integrated Experimental Lab Discovery (GARFIELD), capturing both shared identity and local experimental context through a multi-view experimental protocol graph representation. The graph captures four views: hierarchical, execution, action context, and cross-protocol alignment, making both levels addressable. At inference, GARFIELD agent dynamically interacts with the multi-view experimental protocol graph representation with a dedicated set of tools to adaptively retrieve the necessary context (Figure 1).

We evaluated GARFIELD against two protocol-understanding benchmarks and a protocol-design benchmark using five frontier large language models (LLMs). For protocol understanding, we compare GARFIELD with closed-book prompting and dense retrieval of three or five protocols on tasks covering protocol details retrieval, error detection, resolving procedural ordering, and open-ended questions requiring reasoning over protocols’ structural relationships. GARFIELD consistently improves performance across all tasks and models, outperforming both closed-book and retrieval-based baselines. For protocol design, we evaluate protocol generation under three settings: closed-book LLM, Protocol Context Baseline (PCB), and LLM with GARFIELD context. Across these settings, LLM with GARFIELD context again achieves the highest overall performance. Together, these experiments show that GARFIELD supports both understanding of existing protocols and generating new experimental protocols. In short, our contributions are:

- A Multi-View Experimental Protocol Graph Representation that makes protocol structure, experimental context, and cross-protocol knowledge explicitly accessible for reasoning.
- A Graph-Aware GARFIELD Agent with dedicated tools that dynamically access the graph to retrieve and traverse the context needed for each query.

- A comprehensive evaluation across five frontier LLMs, two protocol-understanding benchmarks, and a protocol-design benchmark, covering protocol detail retrieval, error detection, procedural ordering, structural reasoning, and protocol design.

2 Related Work

Representing experimental procedures. Structured representations turn protocol instructions into explicit actions, participants, and relations. Early ontologies established a vocabulary for this purpose: EXACT formalizes experimental actions and their properties (Soldatova et al., 2008), while SMART Protocols connects a protocol's document structure to its workflow and supports semantic queries over protocol collections (Giraldo et al., 2017). More recent work automates the construction of procedural representations. AutoDSL derives domain-specific languages and constraints from protocol corpora (Shi et al., 2024a), and protocol translation progressively constructs dependence graphs with explicit syntax, semantic detail, and execution links (Shi et al., 2024b). Hierarchically encapsulated representations organize instance actions, generalized operations, and product-flow models at distinct abstraction levels for protocol design (Shi et al., 2025). GARFIELD develops these for protocol understanding, connecting recurring concepts across a collection while keeping each action and experimental object occurrence attached to its own participants, parameters, and procedural context.

Retrieving evidence through graphs. Retrieval-augmented generation gives language models access to external evidence (Lewis et al., 2020). Graph-based approaches extend this access, organizing relationships among the retrieved entities and passages. GraphRAG summarizes communities in an extracted entity graph to answer questions spanning a document collection (Edge et al., 2024). Graph structure can also guide successive retrieval decisions: Think-on-Graph 2.0 alternates graph exploration with document retrieval (Ma et al., 2025), while Youtu-GraphRAG coordinates graph construction and agentic retrieval through a shared schema and hierarchical knowledge organization (Dong et al., 2026). GARFIELD uses connected procedural views to make relevant relation in experimental procedures accessible.

Adaptive retrieval and reasoning. For questions requiring several pieces of evidence, an intermediate result can determine what to retrieve next. IRCoT implements this idea by interleaving retrieval with successive reasoning steps (Trivedi et al., 2023). ReAct extends the interaction to tool actions, allowing returned observations to inform the model's subsequent reasoning and decisions (Yao et al., 2023). Self-RAG learns when to retrieve and how to assess retrieved passages and generated responses through reflection tokens (Asai et al., 2024). These approaches establish evidence gathering as part of the reasoning process. Our agent applies this pattern to protocol graphs, using retrieved occurrences and relations to select the next operation and assemble the evidence needed for an answer.

Scientific tool use and planning. Scientific agents connect language models to specialized computational capabilities. SciToolAgent represents tools in a knowledge graph to support their selection and composition in scientific workflows (Ding et al., 2025). BioProAgent grounds protocol planning in a finite-state machine and uses a design–verify–rectify loop to enforce hardware constraints (Liu et al., 2026). GARFIELD provides structured access to recorded procedural knowledge using a dedicated set of tools, connecting the representation of experimental procedures to question-driven evidence gathering.

3 The Garfield Framework

3.1 Multi-View Experimental Protocol Graph Representation

Let $\mathcal{P} = \{\Phi_1, \ldots, \Phi_N\}$ be a collection of experimental protocols, where each Φ_i specifies a particular experimental procedure. We represent $\mathcal{P}$ through four complementary views,

$$\mathcal{R}(\mathcal{P}) = \langle \mathcal{H}, \mathcal{F}, \mathcal{X}, \mathcal{C} \rangle \,, \tag{1}$$

where $\mathcal{H}$, $\mathcal{F}$, $\mathcal{X}$, and $\mathcal{C}$ correspond to the *Hierarchical View*, *Execution View*, *Action Context View*, and *Cross-Protocol Alignment View*, respectively. Each view is represented as a graph,

$$\mathcal{H} = (\mathcal{V}_H, \mathcal{E}_H), \quad \mathcal{F} = (\mathcal{V}_F, \mathcal{E}_F), \quad \mathcal{X} = (\mathcal{V}_X, \mathcal{E}_X), \quad \mathcal{C} = (\mathcal{V}_C, \mathcal{E}_C), \tag{2}$$

where $\mathcal{V}_H, \mathcal{V}_F, \mathcal{V}_X, \mathcal{V}_C$ are the entities (nodes), and $\mathcal{E}_H, \mathcal{E}_F, \mathcal{E}_X, \mathcal{E}_C$ are the relations in each view (edges). The different views are then connected through shared entities: an entity participating in multiple views is represented by the same node, while each view captures a distinct set of relations involving that entity. Together, the four views form the **Multi-View Experimental Protocol Graph Representation**,

$$\mathcal{G}(\mathcal{P}) = (\mathcal{V}_H \cup \mathcal{V}_F \cup \mathcal{V}_X \cup \mathcal{V}_C,\ \mathcal{E}_H \uplus \mathcal{E}_F \uplus \mathcal{E}_X \uplus \mathcal{E}_C)\,, \tag{3}$$

where $\cup$ is the ordinary union, and $\uplus$ is the disjoint union. We define each view below.

Hierarchical View. Experimental procedures are organized into units at different levels of granularity. We use p for an experimental "process", a higher-level phase directed toward a specific objective; for example, *Precipitate DNA*. We use s for an experimental "step", a procedural unit within a process; for example, *Mix, and store at* $-20°C$ *for at least 1 hour to precipitate the DNA*. We use a for an "atomic action", an individual executable operation within a step; in this example, *mix* and *store* are atomic actions. For protocol Φ_i, the Hierarchical View is $\mathcal{H}_i = (\mathcal{V}_{H_i}, \mathcal{E}_{H_i})$. where $\mathcal{V}_{H_i}$ contains the protocol, process, step, and atomic-action entities,

$$\mathcal{V}_{H_i} = \{\Phi_i\} \cup \mathcal{V}^p_{H_i} \cup \mathcal{V}^s_{H_i} \cup \mathcal{V}^a_{H_i}, \tag{4}$$

and $\mathcal{E}_{H_i}$ contains the hierarchical relations $\Phi_i \to p$, $p \to s$, and $s \to a$, corresponding to protocol-process, process-step, and step-action, respectively.

Execution View. Experimental operations may occur sequentially, in parallel, along alternative paths, or through repetition. These execution patterns are often conveyed by simple textual cues. For example, in *Centrifuge for 15 min. Meanwhile, pre-warm the buffer and label fresh tubes. Then, elute.*, *meanwhile* implies pre-warming the buffer and labeling the tubes can occur in parallel to centrifugation, and *then* indicates that elution follows. The Execution View captures this control-flow structure. For protocol Φ_i, the Execution View is $\mathcal{F}_i = (\mathcal{V}_{F_i}, \mathcal{E}_{F_i})$. The nodes in $\mathcal{V}_{F_i}$ are the process, step, and atomic-action entities shared with the Hierarchical View. The execution relations are,

$$\mathcal{E}_{F_i} = \mathcal{E}^{\mathrm{seq}}_{F_i} \uplus \mathcal{E}^{\mathrm{par}}_{F_i} \uplus \mathcal{E}^{\mathrm{cond}}_{F_i} \uplus \mathcal{E}^{\mathrm{rep}}_{F_i}. \tag{5}$$

Superscripts represent the type of execution: seq (sequential), par (parallel), cond (condition-dependent), rep (repeated).

Action Context View. Atomic actions are defined by the objects they involve and the conditions under which they are performed. For example, in *Centrifuge the DNA lysate in a refrigerated centrifuge at* $12{,}000 \times g$ *for 10 min at* $4°C$, *centrifuge* is the atomic action, *DNA lysate* is the participating object, *refrigerated centrifuge* is the equipment, and the speed, duration, and temperature provide the experimental context. The Action Context View captures these relations. For protocol Φ_i, the Action Context View is $\mathcal{X}_i = (\mathcal{V}_{X_i}, \mathcal{E}_{X_i})$, where

$$\mathcal{V}_{X_i} = \mathcal{V}^a_{X_i} \cup \mathcal{V}^o_{X_i} \cup \mathcal{V}^c_{X_i}. \tag{6}$$

$\mathcal{V}^a_{X_i}$ holds the atomic actions, $\mathcal{V}^o_{X_i}$ stores any related objects (e.g., materials, samples, or equipment), and $\mathcal{V}^c_{X_i}$ preserved context (e.g., parameters, conditions, or rationale). $\mathcal{E}_{X_i}$ links across, connecting actions with their objects and relevant context.

Cross-Protocol Alignment View. Experimental actions and objects may appear in multiple protocols, each time within a different local context. The Cross-Protocol Alignment View introduces a canonical node for each normalized entity and connects it to all corresponding local instances. For example, *centrifuge the DNA lysate at* $12{,}000 \times g$ in one protocol and *centrifuge the bacterial cell suspension at* $5{,}000 \times g$ in another contain distinct local *centrifuge* actions that connect to the same canonical *centrifuge* node. Local instances retain their compositional, execution, and action-context relations, allowing their surrounding context and structure to be compared across protocols. The nodes in this view are,

$$\mathcal{V}_C = \bigcup_{i=1}^{N} \left(\mathcal{V}^a_{X_i} \cup \mathcal{V}^o_{X_i}\right) \cup \mathcal{V}^{c,a}_C \cup \mathcal{V}^{c,o}_C. \tag{7}$$

$\mathcal{V}^{c,a}_C$ and $\mathcal{V}^{c,o}_C$ are the canonical (normalized) action and object nodes, respectively. Edges, $\mathcal{E}_C$, provide the alignment, connecting canonical nodes to local occurrences.

Detailed construction procedures for all views are provided in Appendix A and E.

3.2 GRAPH-AWARE GARFIELD AGENT

The Multi-View Experimental Protocol Graph Representation structures protocol information through hierarchical, execution, action context, and cross-protocol relations. Answering a query now requires locating the relevant graph region, traversing multiple relations, and combining evidence across views. Following recent work on iterative knowledge-graph exploration (Sun et al., 2024) as well as reasoning-and-tool-use formulation (Yao et al., 2023; Gao et al., 2026), we define GARFIELD graph-aware agent, equipped with specialized tools.

The agent receives two components for accessing the graph: a schema card $\mathcal{S}$ (Appendix F, Figure S9) and an interface to the graph-tool library $\mathcal{B} = \{B_1, \ldots, B_J\}$ (Appendix F, Table S14). The schema card describes which entity and relation types exist in the graph and how they can be traversed. This enables the agent to determine where the information required to answer a query is represented and which graph relations or tools are relevant for retrieving it. The graph tools provide the interface to $\mathcal{G}(\mathcal{P})$, allowing the agent to locate concepts, actions, or protocols, inspect local context, follow hierarchical and execution relations, and retrieve source text. In $\mathcal{B}$, each operation is represented by a tool card describing its name, purpose, and arguments.

At reasoning step i, P_i denotes the set of graph tools currently exposed to the agent. Given the query Q, schema card $\mathcal{S}$, current Memory M_{i-1}, and available tools P_i, the agent F_A produces a reasoning thought T_i and a set of tool invocations I_i. If the required capability is not present in P_i, the agent expresses the information need in natural language and invokes ToolRAG, which retrieves the six most relevant tool cards from the full library $\mathcal{B}$; their call specifications then form the tools exposed at the next step. Otherwise, the selected graph tools are executed over $\mathcal{G}(\mathcal{P})$, producing observations E_i. The Memory M_i accumulates the thought T_i, tool invocations I_i, and returned observations E_i, so that subsequent reasoning can build on previously retrieved evidence. This loop continues until the agent outputs STOP and returns the final answer, with inference limited to 16 reasoning steps and 3,000 generated tokens (The full Graph-Aware GARFIELD Agent prompt is available in Appendix F Figure S8). Algorithm 1 summarizes the complete workflow.

Algorithm 1: GARFIELD Agent Workflow

Input: Query Q, $\mathcal{G}(\mathcal{P})$, schema card $\mathcal{S}$, tool library $\mathcal{B}$
Output: Final answer A

Initialize $M_0 \leftarrow \emptyset$, $P_1 = \emptyset$, $i \leftarrow 0$;
while *reasoning is incomplete* **do**
  $i \leftarrow i + 1$;
  Generate thought: $T_i \leftarrow F_A(Q, M_{i-1}, \mathcal{S}, P_i)$;
  Generate tool calls: $I_i \leftarrow F_A(Q, M_{i-1}, T_i, \mathcal{S}, P_i)$;
  if $T_i =$ STOP **then**
    return *Final Answer*;
  else
    if TOOLRAG $\in I_i$ **then**
      $E_i \leftarrow$ output of TOOLRAG execution;
      $P_{i+1} \leftarrow E_i$;
    else
      Execute graph tools from I_i over $\mathcal{G}(\mathcal{P})$ to obtain E_i;
      $P_{i+1} \leftarrow P_i$;
    Observe E_i;
    $M_i \leftarrow M_{i-1} \cup \{T_i, I_i, E_i\}$;

4 EXPERIMENTAL SETUP

Benchmarks & Baselines. To evaluate GARFIELD, we use two challenging benchmarks. (1) **BioPIE** (Hou et al., 2026) with 3,527 protocols from the journals Cell/STAR Protocols, Nature, and JoVE, with 3,275 open-ended QA that aim to understand the structure of the experimental protocols. Reasoning depth is measured by graph hops: 1-hop questions follow a direct relation, and multi-hop questions traverse a chain of relations. The benchmark has 2,935 single-hop, 285 double-hop, and 55 triple-or-higher-hop questions. (2) From **BioProBench** (Liu et al., 2025) we use 9,374 protocols

from protocols.io, with 6,000 questions in three tasks: Protocol Question Answering (PQA), Error Correction (ERR), and Step Ordering (ORD). PQA tests correct retrieval of protocol details, ERR tests error detection, and ORD tests the ability to reconstruct an ordering and is separated into child-step and top-step cases. Child-step cases order detailed steps within a section, whereas top-step cases order the protocol's major processes. To evaluate performance on protocol design, we used the test set from a benchmark presented by Shi et al. (2025).

Evaluation Metrics. For open-ended questions, we report Exact Match (EM), token-level F1, and LLM-as-a-judge scores using Opus-4.8 and GPT-5.4 as evaluators (human validation of the LLM-as-a-judge evaluation is provided in Appendix C, Table S10). For multiple-choice, error detection, and ordering tasks, we report accuracy (Acc.), F1, precision (Prec.), Sequence Exact Match (Seq-EM), pairwise agreement (PW), and Kendall's τ(Xu et al., 2026; Liu et al., 2025; Rajpurkar et al., 2016). For protocol design evaluation, following prior work (Shi et al., 2025), we report intersection over union (IoU) for operations, $\mathrm{IoU(Op)}$; materials, biosamples and intermediate products, $\mathrm{IoU(Prod)}$; and devices, $\mathrm{IoU(Dev)}$. We additionally report similarity between execution sequences, $\mathrm{Sim(Exec)}$, and similarity between complete protocols at the parameter level, $\mathrm{Sim(Param)}$.

Implementation Details. Experiments were performed using Claude Opus-4.8, GPT-5.4, GPT-5.6-Sol, Claude Sonnet 5, and Claude Haiku 4.5, in four experimental settings. **closed-book LLM** answers each query without external context. **retrieving@3** and **retrieving@5** embed the query and protocol corpus using `text-embedding-3-large` and retrieve the top-k protocols as context for generation, with $k = 3$ and $k = 5$, respectively. **GARFIELD** inputs the query to the agent, which accesses the protocol graph through its tools. Agent execution is limited to 3,000 generated tokens and 16 steps, with each ToolRAG call exposing the top-6 tools ranked as most relevant to the current agent need. Graph construction for each benchmark is performed once, offline during ingestion.

For protocol design, the model receives a target protocol title and a short AI-generated description available in the ground truth, with varying level of added context: 1) closed-book LLM, no external context; 2) **Protocol Context Baseline (PCB)**, providing three relevant protocols from the original corpus given as unstructured textual context; and 3) **LLM with GARFIELD context**, where six protocols are given in the GARFIELD structured schema; the same three previously retrieved and three additional protocols, selected from the top-100 retrieved neighborhood based on overlap in actions, materials, and instruments. The model is provided with shared canonical actions, materials, and instruments from the top neighborhood, together with their contextual structure.

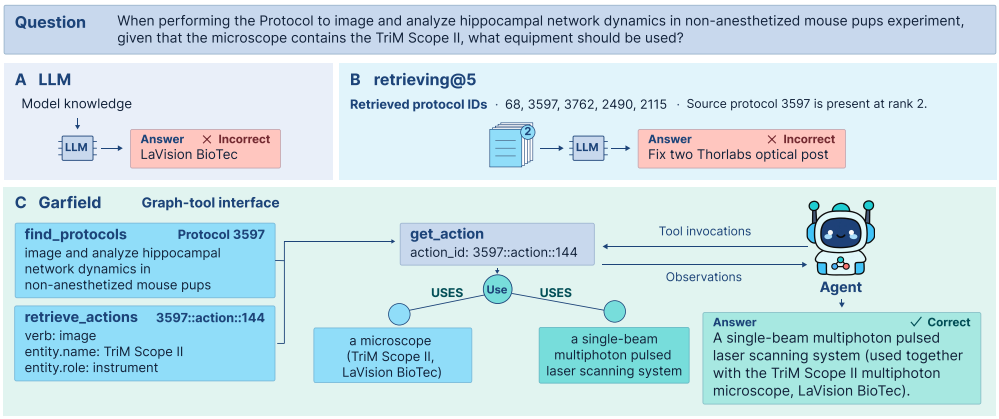

Figure 2: Example of open-ended protocol question answering comparing three settings: a) closed-book LLM, b) retrieving@5, and c) GARFIELD.

5 RESULTS

5.1 PROTOCOL DETAILS, ERROR DETECTION, AND ORDERING

Across BioProBench tasks, GARFIELD consistently outperforms closed-book LLM and retrieval-based baselines across all five LLMs (Table 1). GARFIELD improves protocol understanding across

question answering, error detection, and procedural ordering. Averaged across the LLM models, GARFIELD improves PQA accuracy by 25.4% over closed-book LLM and 11.5% over retrieving@5. For ERR, accuracy improves by 19.5% over closed-book LLM and 11.6% over retrieving@5, while F1 improves by 0.196 and 0.108, respectively. For ORD, sequence EM improves by 25.6% over closed-book LLM and 14.4% over retrieving@5, while Kendall's τ improves by 0.148 and 0.068, respectively. Ordering is the most difficult task, with the highest sequence EM reaching 66.0%. However, GARFIELD achieves PW between 81.1% and 90.0% across different LLMs (Appendix B.1), showing that many predictions preserve correct relative order of steps, even when full sequence is not exactly matched. In addition to these results, Appendix B.2 reports domain-specific results across experimental protocol domains. Tool-call distribution of GARFIELD Agent is reported in Appendix D.

Results on ordering sequence EM on child steps and top steps emphasize the challenge of this task (Table 2). Dense text retrieval improves child-step sequence EM from 42.04% to 61.64% when moving from LLM to retrieving@5. Top step EM is essentially unchanged, at 25.68% versus 25.70%. Increasing access to relevant text is therefore associated with a large gain in local ordering but little change in exact higher-level ordering in these runs. GARFIELD improves both, child-step EM reaches 75.10%, and top step EM reaches 41.38%. The gains over retrieving@5 are 13.46 and 15.68%. The top step gain is also accompanied by improvements of 4.98% in pairwise agreement and 0.100 in Kendall's τ. This shows that retrieving relevant text is not enough to capture how a protocol is organized. By structuring the same protocol through multiple graph views, GARFIELD can find relationships between steps, and the broader process, leading to better ordering at all levels.

Table 1: BioProBench performance on PQA, ERR, and ORD across five backbone LLMs under four settings: closed-book LLM, retrieving@3, retrieving@5, and GARFIELD. The best result within each backbone LLM is shown in **bold**.

		PQA	**ERR**		**ORD**	
Model	Setting	Acc. (%)	Acc. (%)	F1	Seq-EM (%)	τ
Opus-4.8	closed-book LLM	71.0	69.0	0.62	37.3	0.60
	retrieving@3	83.7	73.1	0.70	49.5	0.69
	retrieving@5	85.1	76.3	0.73	49.3	0.69
	GARFIELD	**93.5**	**87.4**	**0.82**	**60.8**	**0.74**
GPT-5.4	closed-book LLM	70.0	67.8	0.59	35.6	0.61
	retrieving@3	81.6	72.4	0.69	47.3	0.68
	retrieving@5	82.6	75.4	0.72	46.7	0.67
	GARFIELD	**93.2**	**82.4**	**0.78**	**62.9**	**0.76**
GPT-5.6 Sol	closed-book LLM	74.5	73.2	0.66	40.7	0.64
	retrieving@3	85.5	72.5	0.69	50.2	0.69
	retrieving@5	86.2	76.3	0.73	50.4	0.70
	GARFIELD	**97.2**	**89.6**	**0.86**	**66.0**	**0.80**
Sonnet-5	closed-book LLM	69.5	69.7	0.64	34.2	0.52
	retrieving@3	82.7	72.6	0.65	47.1	0.60
	retrieving@5	83.4	76.2	0.68	47.3	0.61
	GARFIELD	**95.3**	**88.0**	**0.83**	**59.3**	**0.65**
Haiku-4.5	closed-book LLM	60.6	53.0	0.59	27.5	0.46
	retrieving@3	77.6	66.0	0.67	37.0	0.53
	retrieving@5	77.7	67.8	0.68	37.7	0.56
	GARFIELD	**93.3**	**82.6**	**0.79**	**54.4**	**0.62**

5.2 OPEN-ENDED PROTOCOL QUESTION ANSWERING

GARFIELD's structured context also supports open-ended protocol question answering (Table 3). Averaged across the five LLMs, EM increases from 10.76% for the baseline LLM, 21.18% for retrieving@3, and 25.64% for retrieving@5 to 58.36% for GARFIELD, showing greater agreement with the reference answers. Relative to retrieving@5, GARFIELD improves mean Token-F1 by 0.326. The judge scores show similar gains, increasing by 27.3% for the Opus-4.8 judge and 24.4% for the

Table 2: BioProBench ordering Seq-EM(%) across child-step and top-step ordering, averaged over five backbone LLMs under four settings. The best result in each column is shown in **bold**.

Setting	Child steps	Top steps	Overall
closed-book LLM	42.04	25.68	35.06
retrieving@3	61.64	25.54	46.22
retrieving@5	61.64	25.70	46.28
GARFIELD	**75.10**	**41.38**	**60.68**

Table 3: BioPIE open-ended question answering results on all 3,275 questions across five backbone LLMs under four settings. J_O and J_G use Opus-4.8 and GPT-5.4 as judges, respectively. The best result within each backbone LLM is shown in **bold**.

Model	Setting	EM (%)	Token-F1	J_O (%)	J_G (%)
Opus-4.8	closed-book LLM	12.9	0.279	36.3	39.1
	retrieving@3	30.2	0.475	57.6	60.4
	retrieving@5	36.0	0.490	59.0	60.2
	GARFIELD	**58.3**	**0.739**	**83.0**	**82.0**
GPT-5.4	closed-book LLM	10.7	0.266	32.3	36.5
	retrieving@3	20.9	0.440	56.4	59.3
	retrieving@5	29.3	0.469	58.3	60.7
	GARFIELD	**59.8**	**0.749**	**83.7**	**82.1**
GPT-5.6-sol	closed-book LLM	12.0	0.282	35.8	39.1
	retrieving@3	13.3	0.371	58.5	61.4
	retrieving@5	18.7	0.400	60.8	62.6
	GARFIELD	**59.2**	**0.754**	**85.7**	**84.6**
Sonnet-5	closed-book LLM	10.3	0.251	34.0	36.2
	retrieving@3	28.8	0.453	55.9	58.2
	retrieving@5	30.0	0.447	56.0	57.4
	GARFIELD	**58.2**	**0.743**	**84.3**	**83.2**
Haiku-4.5	closed-book LLM	7.9	0.184	26.2	29.1
	retrieving@3	12.7	0.245	46.7	47.7
	retrieving@5	14.2	0.254	45.4	47.3
	GARFIELD	**56.3**	**0.704**	**79.2**	**78.3**

GPT-5.4 judge. Together, the results show that GARFIELD consistently improves answer agreement and correctness across all five LLMs for open-ended protocol questions.

Table 4 shows that GARFIELD's EM gains persist across reasoning depths. Averaged across the five LLMs within each depth group, EM gains are 33.0, 30.2, and 28.7% for single-, double-, and triple+-hop questions, respectively. Mean Token-F1 gains are 0.329, 0.310, and 0.277. Averaged across both judges and all five LLMs, judge-score gains are 26.1, 24.4, and 18.7%, respectively. Full results for each reasoning-depth subset and tool-call distributions are reported in Appendix C and D, respectively. We also show an example in Figure 2 where GARFIELD identifies the correct equipment, while the closed-book LLM and retrieving@5 answer incorrectly even though retrieving@5 retrieves the source protocol.

5.3 PROTOCOL DESIGN

Evaluation on design tasks shows that adding protocol context improves generation across all models, with GARFIELD giving the strongest results (Table 5). Averaged across models, IoU(Op), IoU(Prod), IoU(Dev) increase from 0.172, 0.133, and 0.129 with closed-book LLM to 0.203, 0.164, and 0.139 with PCB, and further to 0.258, 0.200, and 0.174 with GARFIELD. Sim(Exec) follows the same

Table 4: BioPIE question-answering performance across reasoning depths (single-, double-, and triple+-hop questions), measured by EM (%) and averaged over five backbone LLMs under four settings. The best result within each backbone LLM is shown in **bold**.

Depth	Questions	closed-book LLM	retrieving@3	retrieving@5	GARFIELD
Single	2,935	10.66	21.39	25.86	**58.88**
Double	285	11.23	19.95	23.72	**53.96**
Triple+	55	14.20	17.83	24.72	**53.44**

trend, with average $\mathrm{Sim(Exec)}$ increasing from 0.306 to 0.348 and 0.396 for closed-book LLM, PCB, and GARFIELD, respectively.Sim(Param) changes from 0.802 to 0.809 and 0.822. Overall, GARFIELD achieves the best performance across nearly all models and evaluation metrics.

Table 5: Evaluation of protocol design task across five backbone LLMs under three settings: closed-book LLM, PCB, and GARFIELD. The best result within each backbone LLM is shown in **bold**.

Model	Setting	IoU(Op)	IoU(Prod)	IoU(Dev)	Sim(Param)	Sim(Exec)
Opus-4.8	closed-book LLM	0.180	0.138	0.139	0.818	0.308
	PCB	0.200	0.166	0.143	0.818	0.342
	GARFIELD	**0.250**	**0.196**	**0.183**	**0.840**	**0.388**
GPT-5.4	closed-book LLM	0.169	0.141	0.127	0.807	0.302
	PCB	0.220	0.184	0.142	**0.813**	0.361
	GARFIELD	**0.286**	**0.221**	**0.167**	0.807	**0.408**
GPT-5.6-sol	closed-book LLM	0.148	0.114	0.112	0.792	0.283
	PCB	0.194	0.149	0.131	0.797	0.350
	GARFIELD	**0.240**	**0.186**	**0.161**	**0.815**	**0.401**
Sonnet-5	closed-book LLM	0.170	0.119	0.135	0.809	0.293
	PCB	0.198	0.160	0.144	0.818	0.333
	GARFIELD	**0.256**	**0.207**	**0.183**	**0.839**	**0.386**
Haiku-4.5	closed-book LLM	0.193	0.154	0.132	0.785	0.345
	PCB	0.202	0.162	0.136	0.798	0.355
	GARFIELD	**0.256**	**0.188**	**0.177**	**0.811**	**0.395**

6 CONCLUSION AND DISCUSSION

We introduced GARFIELD, a framework that represents experimental protocols as structured context through a multi-view experimental protocol graph and enables a graph-aware agent to adaptively access this structure using dedicated tools. Across protocol understanding tasks, GARFIELD consistently outperforms language-model and retrieval baselines, improving retrieval of experimental details by 11.5% accuracy, reasoning over structural relationships by 0.326 Token-F1, error detection by 11.6% accuracy, and procedural ordering by 14.4% seq-EM compared with retrieving@5. GARFIELD also improves protocol design, increasing IoU(Op) and Sim(Exec) by 27.0% and 13.6%, respectively, over the protocol-context baseline (PCB). For GARFIELD 's limitations and future directions, the current retrieval evaluation remains in-distribution, leaving generalization to unseen protocols and settings without access to the original source protocol untested. Expanding the graph and developing dedicated domain-specific corpora could enable this form of generalization and further extend the utility of the framework. For protocol design, future work should include expert review and laboratory validation, since computational metrics alone cannot determine whether a generated protocol is experimentally practical to execute. Overall, GARFIELD provides a structured foundation for language models to more effectively use experimental knowledge for both protocol understanding and design.

AI USE STATEMENT

We used LLMs (e.g., GPT-5.6 Sol and Grammarly) to assist in polishing the writing of this paper. Specifically, LLMs were used to refine grammar, improve fluency, and adjust the tone of the academic writing under strict human supervision. Generative AI was also used for limited assistance with code syntax, bug fixing, and code organization, under strict human supervision and step-by-step review. We did not use generative AI tools for data generation; developing theoretical models or conceptual frameworks; designing or providing feedback on research methodology or experiments; translation; cleaning or reformatting datasets; supporting qualitative or thematic data analysis; or interpreting results.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *International Conference on Learning Representations*, volume 2024, pp. 9112–9141, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/25f7be9694d7b32d5cc670927b8091e1-Paper-Conference.pdf.

Wenduo Cheng, Mingqian Ma, Shuaike Shen, Anna Hupalowska, Jennifer E. Rood, Yang Zhang, Gaurav Agrawal, Christine Bakan, Michelle A. Lee, Aviv Regev, and Jian Ma. Towards human-led, agent-driven autonomous laboratories for the life sciences, August 2026. URL https://doi.org/10.20944/preprints202608.0273.v1.

Keyan Ding, Jing Yu, Junjie Huang, Yuchen Yang, Qiang Zhang, and Huajun Chen. SciToolAgent: a knowledge-graph-driven scientific agent for multitool integration. *Nature Computational Science*, 5(10):962–972, 2025. doi: 10.1038/s43588-025-00849-y. URL https://doi.org/10.1038/s43588-025-00849-y.

Junnan Dong, Siyu An, Yifei Yu, Qian-Wen Zhang, Linhao Luo, Xiao Huang, Di Yin, Yunsheng Wu, and Xing Sun. Youtu-GraphRAG: Vertically Unified Agents for Graph Retrieval-Augmented Complex Reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 143735–143755, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e8618f2038b0209705e4d5e3cd496e1d-Paper-Conference.pdf.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv, 2024. URL https://arxiv.org/abs/2404.16130.

Shanghua Gao, Ayush Noori, Richard Zhu, Curtis Ginder, Zhenglun Kong, Xiaorui Su, Justin Kauffman, Benjamin S. Glicksberg, Joshua Lampert, Ankit Sakhuja, Ashwin Sawant, ATHENA-R1 Evaluation Consortium, David A. Clifton, Noa Dagan, Ran Balicer, and Marinka Zitnik. An AI agent for treatment reasoning over a biomedical tool universe. arXiv, 2026. URL https://arxiv.org/abs/2606.28692.

Olga Giraldo, Alexander García, Federico López, and Oscar Corcho. Using semantics for representing experimental protocols. *Journal of Biomedical Semantics*, 8(1):52, 2017. doi: 10.1186/s13326-017-0160-y. URL https://doi.org/10.1186/s13326-017-0160-y.

Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Petar Sirkovic, Artiom Myaskovsky, Grzegorz Glowaty, Felix Weissenberger, Alessio Orlandi, Dan Popovici, Anil Palepu, Keran Rong, Ryutaro Tanno, Khaled Saab, Fan Zhang, Jacob Blum, Andrew Carroll, Kavita Kulkarni, Nenad Tomašev, Dina Zverinski, Ivor Rendulic, Elahe Vedadi, Florian Hasler, Luka Rimanic, Marina Boia, Ivan Budiselic, Ben Feinstein, Mathias Bellaiche, Tom Sheffer, Jan Freyberg, Jeremy Ratcliff, Ottavia Bertolli, Katherine Chou, Avinatan Hassidim, Burak Gokturk, Amin Vahdat, Yuan Guan, Vikram Dhillon, Eeshit Dhaval Vaishnav, Byron Lee, Tiago R. D. Costa, José R. Penadés, Gary Peltz, Yossi Matias, James Manyika, Demis Hassabis, Yunhan Xu, Pushmeet Kohli, Annalisa Pawlosky, Alan Karthikesalingam, and Vivek Natarajan. Accelerating scientific discovery with Co-Scientist. *Nature*, 655(8122):487–496, 2026. doi: 10.1038/s41586-026-10644-y. URL https://www.nature.com/articles/s41586-026-10644-y.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to Memory: Non-Parametric Continual Learning for Large Language Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Haofei Hou, Shunyi Zhao, Fanxu Meng, Kairui Yang, Lecheng Ruan, and Qining Wang. BioPIE: A biomedical protocol information extraction dataset for experiment understanding. arXiv, 2026. URL https://arxiv.org/abs/2601.04524v3.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Ryan Li, Yusuf Roohani, Lin Qiu, Shiyi Cao, Gavin Li, Junze Zhang, Di Yin, Rick Wierenga, Deniz Kavi, Sherry Liu, Tianwei She, Shruti Marwaha, Jennefer N. Carter, Xin Zhou, Matthew T. Wheeler, Jonathan A. Bernstein, Mengdi Wang, Peng He, Jingtian Zhou, Michael P. Snyder, Le Cong, Aviv Regev, and Jure Leskovec. Autonomous biomedical research with an artificial intelligence agent. *Science*, 393 (6813):eadz4351, 2026. doi: 10.1126/science.adz4351. URL https://doi.org/10.1126/science.adz4351.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Yuyang Liu, Liuzhenghao Lv, Xiancheng Zhang, Li Yuan, and Yonghong Tian. BioProBench: Comprehensive dataset and benchmark in biological protocol understanding and reasoning. arXiv, 2025. URL https://arxiv.org/abs/2505.07889v1.

Yuyang Liu, Jingya Wang, Liuzhenghao Lv, and Yonghong Tian. BioProAgent: Neuro-Symbolic Grounding for Constrained Scientific Planning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 42764–42783, 2026. doi: 10.18653/v1/2026.acl-long.1981. URL https://aclanthology.org/2026.acl-long.1981/.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-Graph 2.0: Deep and Faithful Large Language Model Reasoning with Knowledge-guided Retrieval Augmented Generation. In *International Conference on Learning Representations*, volume 2025, pp. 52782–52806, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/830b1abc6d2da85f23d41169fa44d185-Paper-Conference.pdf.

James McLaughlin, Josh Lagrimas, Haider Iqbal, Helen Parkinson, and Henriette Harmse. Ols4: a new ontology lookup service for a growing interdisciplinary knowledge ecosystem. *Bioinformatics*, 41(5):btaf279, 2025.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016. doi: 10.18653/v1/d16-1264. URL https://doi.org/10.18653/v1/d16-1264.

Yu-Zhe Shi, Haofei Hou, Zhangqian Bi, Fanxu Meng, Xiang Wei, Lecheng Ruan, and Qining Wang. AutoDSL: Automated domain-specific language design for structural representation of procedures with constraints. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12177–12214, 2024a. doi: 10.18653/v1/2024.acl-long.659. URL https://aclanthology.org/2024.acl-long.659/.

Yu-Zhe Shi, Fanxu Meng, Haofei Hou, Zhangqian Bi, Qiao Xu, Lecheng Ruan, and Qining Wang. Expert-level protocol translation for self-driving labs. In *Advances in Neural Information Processing Systems*, volume 37, pp. 47488–47529, 2024b. doi: 10.52202/079017-1506. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/54dd9e0cff6d9214e20d97eb2a3bae49-Paper-Conference.pdf.

Yu-Zhe Shi, Mingchen Liu, Fanxu Meng, Qiao Xu, Zhangqian Bi, Kun He, Lecheng Ruan, and Qining Wang. Hierarchically Encapsulated Representation for Protocol Design in Self-Driving Labs. In *International Conference on Learning Representations*, volume 2025, pp. 89146–89195, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/dd9b76f050a86a3ded6135ad3556e786-Paper-Conference.pdf.

Aditya Sivakumar, Ashu Singhal, Nicholas Larus-Stone, and Nithin Parsan. Benchbench-protocol: Evaluating real-world wet-lab protocol reasoning and modification. *arXiv preprint arXiv:2608.23898*, 2026. doi: 10.48550/arXiv.2608.23898. URL https://doi.org/10.48550/arXiv.2608.23898.

Larisa N Soldatova, Wayne Aubrey, Ross D King, and Amanda Clare. The EXACT description of biomedical protocols. *Bioinformatics*, 24(13):i295–i303, 2008. doi: 10.1093/bioinformatics/btn156. URL https://doi.org/10.1093/bioinformatics/btn156.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph. In *International Conference on Learning Representations*, volume 2024, pp. 3868–3898, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/10a6bdcabbd5a3d36b760daa295f63c1-Paper-Conference.pdf.

Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 646(8085):716–723, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09442-9. URL https://doi.org/10.1038/s41586-025-09442-9.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Wanghan Xu, Xiangyu Zhao, Yuhao Zhou, Xiaoyu Yue, Ben Fei, Fenghua Ling, Wenlong Zhang, and Lei Bai. EarthSE: A benchmark evaluating earth scientific exploration capability for large language models. In *International Conference on Learning Representations*, volume 2026, pp. 130942–130966, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/d4ce6738e84876aa79f13c8bc8b7c5eb-Abstract-Conference.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

A Multi-View Experimental Protocol Graph Construction

A.1 Hierarchical View Extraction

The Hierarchical View is constructed by decomposing each protocol into four levels: Protocol → Process → Step → Action. Protocols are provided either as pre-numbered lines or as continuous text. Pre-numbered protocols are passed to the extractor using their original line numbers as source references, while continuous text is segmented using the `spaCy` rule-based Sentencizer and the resulting sentences are numbered sequentially. Each numbered line or segmented sentence is treated as a source unit, giving both input formats a common ordered representation before extraction. Opus-4.8 is used with the Hierarchical View Extractor Prompt (Figure S3) to extract the Hierarchical View from the indexed source protocols. For each Process, the model either uses an existing heading as its title or generates a short title when no heading is available. Each Step is derived from either a single source unit, a merge of consecutive units, or a split of one unit containing multiple independent instructions. Each resulting Step is then decomposed into Actions, each represented by its surface verb and a verbatim source span. The model predicts only the hierarchical structure and does not rewrite Step or Action content. Step text is reconstructed directly from the referenced source units, keeping the extracted hierarchy grounded in the original protocol. Each predicted hierarchy is validated using deterministic structural and string-matching checks. These verify complete and ordered source-unit coverage, contiguous merged Steps and Process spans, verbatim split spans, and exact occurrence of each Action span within its parent Step.

A.2 Execution View Extraction

Before Execution View extraction, Opus-4.8 is used with the Preprocessor Prompt (Figure S4) to resolve unambiguous anaphoric references and expand confidently identified abbreviations. The preprocessor changes only the referenced terms or abbreviations and preserves all remaining Action text verbatim. This step ensures that later extractors receive the exact entities and technical terms referred to by each Action, rather than unresolved pronouns or abbreviated forms.

The Execution View is constructed over the Process, Step, and Action entities produced by the Hierarchical View. Sequential `NEXT` relations are added deterministically from their hierarchical order, connecting consecutive Processes, consecutive Steps within each Process, and consecutive Actions within each Step. The Execution View Extractor Prompt (Figure S5) is then applied to identify non-sequential execution structures explicitly stated in the protocol: repetition, condition-dependent execution, optional or alternative paths, and parallel execution. For each detected structure, the extractor returns the affected Process, Step, or Action indices together with a verbatim evidence span from the protocol. Repetition and parallelism are represented at the hierarchy level specified by the source text, while branching introduces a Decision node connected to the corresponding execution outcomes. Repetition is encoded by a `REPEATS` edge from the end of the repeated block to its beginning, and concurrent elements are connected by `OVERLAPS_WITH`. Extracted structures are validated by checking that all referenced entities exist and that each evidence span occurs verbatim in the protocol before the corresponding execution relations are added to the graph.

A.3 Action Context View Extraction

The Action Context View is constructed over the Atomic Actions produced by the Hierarchical View after preprocessing. Opus-4.8 is used with the Action Context View Extractor Prompt (Figure S6) to generate one structured record for each Action. For each Action, the extractor identifies experimental objects (materials, biological samples, instruments, and outputs), together with parameters, locations, methods, modifiers, required conditions, rationale, and recipes. Each extracted field is grounded by a verbatim evidence span from the Action text. Protocol-level reference identifiers are assigned to experimental objects and locations; repeated mentions of the same entity reuse the same identifier, while distinct entities receive separate identifiers. Only information explicitly stated in the Action text is extracted; missing entities, parameters, conditions, or rationale are not inferred.

The extracted records are converted into the entities and relations defined in the Action Context View. Actions are connected to experimental objects through `USES` and `PRODUCES`, while contextual information is represented through `HAS_PARAMETER`, `USES_METHOD`, `SITE`, `LOCATED_AT`, `HAS_MODIFIER`, `HAS_RATIONALE`, `HAS_REQUIRED_CONDITION`,

HAS_RECIPE, and HAS_PROPERTY. After extraction, each Action Context record is verified using deterministic string matching against the protocol text. Extracted entities and parameter values are retained only when supported by the source, while modifiers, required conditions, rationale, and properties require a verbatim evidence span. Unsupported fields are removed; outputs are exempt from direct name matching.

A.4 Cross-Protocol Alignment View Extraction

The Cross-Protocol Alignment View is constructed from the local Actions and experimental objects extracted across all protocols. The Entity Normalization Prompt (Figure S7) is applied to both Actions and experimental objects to produce a canonical name and an immediate broader parent category while preserving protocol-specific context on the local instance. For Actions, surface and grammatical variations are mapped to the underlying operation; for experimental objects, instance-specific details such as quantities, concentrations, temperatures, and handling conditions are removed while identity-defining information is retained.

Canonical Actions, Materials, Biological Samples, Instruments, and Outputs are maintained separately by type. For each normalized instance, its canonical name is first checked for an exact name match against existing canonical nodes of the same type. If no exact match is found, the name is queried through the Ontology Lookup Service (McLaughlin et al., 2025) to obtain a standard ontology identifier when available, and is then compared with the three most similar existing canonical nodes of the same type using embeddings and a strict identity check. If a candidate is accepted, the local instance is linked to the existing canonical node through INSTANCE_OF; otherwise, a new canonical node is created. Each canonical node stores its parent category, an ontology identifier when available, and a textual description retrieved from the Wikipedia page-summary endpoint through the Wikimedia REST API.

In addition to the deterministic structural and string-matching validation procedures applied during graph construction, we manually verified the extracted views for 20 randomly sampled protocols against the corresponding source protocols, confirming that the hierarchical structure, execution relations, action context, and cross-protocol alignments were correctly captured without introducing unsupported experimental information or omitting relevant procedural details.

B COMPREHENSIVE BIOPROBENCH RESULTS

This section provides the complete BioProBench evaluation results. Section B.1 reports aggregate performance across the full benchmark for PQA, ERR, and ORD under all five backbone LLMs and four experimental settings. Section B.2 provides a domain-specific breakdown of the same evaluation across eight experimental domains, showing how performance varies across different biological fields.

B.1 AGGREGATE BIOPROBENCH RESULTS

Table S1: Detailed BioProBench results on PQA, ERR, and ORD across five backbone LLMs under four settings: closed-book LLM, retrieving@3, retrieving@5, and GARFIELD. The best result within each backbone LLM is shown in **bold**.

		PQA	**ERR**			**ORD**								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	71.0	69.0	0.58	0.62	45.1	84.9	+0.70	26.9	73.5	+0.47	37.3	80.0	+0.60
	retrieving@3	83.7	73.1	0.61	0.70	65.3	91.2	+0.82	28.2	75.4	+0.51	49.5	84.5	+0.69
	retrieving@5	85.1	76.3	0.65	0.73	64.8	91.3	+0.83	28.5	74.9	+0.50	49.3	84.3	+0.69
	GARFIELD	**93.5**	**87.4**	**0.91**	**0.82**	**74.0**	**91.8**	**+0.84**	**43.0**	**80.4**	**+0.61**	**60.8**	**86.9**	**+0.74**
GPT-5.4	closed-book LLM	70.0	67.8	0.57	0.59	42.7	85.1	+0.70	26.2	74.5	+0.49	35.6	80.6	+0.61
	retrieving@3	81.6	72.4	0.60	0.69	62.5	90.6	+0.81	26.9	74.7	+0.49	47.3	83.8	+0.68
	retrieving@5	82.6	75.4	0.64	0.72	62.3	90.4	+0.81	25.7	74.6	+0.49	46.7	83.7	+0.67
	GARFIELD	**93.2**	**82.4**	**0.74**	**0.78**	**77.6**	**94.0**	**+0.88**	**43.3**	**80.5**	**+0.61**	**62.9**	**88.2**	**+0.76**
GPT-5.6-sol	closed-book LLM	74.5	73.2	0.64	0.66	49.2	87.0	+0.74	29.4	75.5	+0.51	40.7	82.1	+0.64
	retrieving@3	85.5	72.5	0.60	0.69	66.3	91.7	+0.83	28.7	75.3	+0.51	50.2	84.7	+0.69
	retrieving@5	86.2	76.3	0.65	0.73	65.9	91.5	+0.83	29.7	76.0	+0.52	50.4	84.9	+0.70
	GARFIELD	**97.2**	**89.6**	**0.90**	**0.86**	**80.4**	**95.2**	**+0.90**	**46.8**	**83.2**	**+0.66**	**66.0**	**90.0**	**+0.80**
Sonnet-5	closed-book LLM	69.5	69.7	0.59	0.64	40.4	80.6	+0.61	25.7	70.1	+0.40	34.2	76.1	+0.52
	retrieving@3	82.7	72.6	0.64	0.65	62.5	87.1	+0.74	26.5	70.3	+0.41	47.1	79.9	+0.60
	retrieving@5	83.4	76.2	0.70	0.68	62.5	88.0	+0.76	26.9	70.5	+0.41	47.3	80.6	+0.61
	GARFIELD	**95.3**	**88.0**	**0.91**	**0.83**	**74.3**	**89.7**	**+0.79**	**39.2**	**73.5**	**+0.47**	**59.3**	**82.7**	**+0.65**
Haiku-4.5	closed-book LLM	60.6	53.0	0.44	0.59	32.8	81.3	+0.63	20.2	62.4	+0.25	27.5	73.2	+0.46
	retrieving@3	77.6	66.0	0.53	0.67	51.6	87.0	+0.74	17.4	62.8	+0.26	37.0	76.7	+0.53
	retrieving@5	77.7	67.8	0.55	0.68	52.7	87.7	+0.75	17.7	65.2	+0.30	37.7	78.1	+0.56
	GARFIELD	**93.3**	**82.6**	**0.73**	**0.79**	**69.2**	**90.5**	**+0.81**	**34.6**	**68.5**	**+0.37**	**54.4**	**81.1**	**+0.62**

B.2 DOMAIN-SPECIFIC BIOPROBENCH RESULTS

Table S2: BioProBench results for questions in the *Molecular Biology Techniques* domain, with n_{PQA}=386, n_{ERR}=201, and n_{ORD}=274. Results are reported for five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result for each LLM is shown in **bold**.

		PQA	**ERR**			**ORD**								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	72.5	69.2	0.55	0.60	49.7	88.8	+0.78	15.7	69.4	+0.39	38.7	82.5	+0.65
	retrieving@3	84.5	74.1	0.60	0.68	64.9	91.8	+0.84	21.3	73.1	+0.46	50.7	85.7	+0.71
	retrieving@5	84.5	76.1	0.63	0.69	64.3	92.4	+0.85	20.2	71.9	+0.44	50.0	85.7	+0.71
	GARFIELD	**93.5**	**82.1**	**0.81**	**0.71**	**77.8**	**94.1**	**+0.88**	**32.6**	**78.1**	**+0.56**	**63.1**	**88.9**	**+0.78**
GPT-5.4	closed-book LLM	69.7	70.1	0.56	0.59	45.9	87.3	+0.75	16.9	71.4	+0.43	36.5	82.1	+0.64
	retrieving@3	79.8	72.6	0.58	0.66	58.9	89.9	+0.80	15.7	71.8	+0.44	44.9	84.0	+0.68
	retrieving@5	81.6	74.1	0.60	0.67	61.6	91.2	+0.82	15.7	72.2	+0.44	46.7	85.0	+0.70
	GARFIELD	**91.5**	**75.1**	**0.62**	**0.67**	**81.1**	**96.5**	**+0.93**	**32.6**	**79.3**	**+0.59**	**65.3**	**90.9**	**+0.82**
GPT-5.6-sol	closed-book LLM	75.4	75.6	0.63	0.67	53.5	89.3	+0.79	19.1	71.3	+0.43	42.3	83.5	+0.67
	retrieving@3	86.0	74.6	0.60	0.69	64.9	91.5	+0.83	19.1	72.5	+0.45	50.0	85.3	+0.71
	retrieving@5	86.3	74.6	0.60	0.70	63.2	92.1	+0.84	22.5	73.4	+0.47	50.0	86.0	+0.72
	GARFIELD	**96.4**	**85.6**	**0.87**	**0.77**	**82.7**	**96.4**	**+0.93**	**33.7**	**79.9**	**+0.60**	**66.8**	**91.1**	**+0.82**
Sonnet-5	closed-book LLM	69.2	69.2	0.55	0.62	42.2	84.2	+0.68	14.6	69.5	+0.39	33.2	79.4	+0.59
	retrieving@3	82.4	77.1	0.68	0.68	62.7	88.0	+0.76	15.7	68.5	+0.37	47.4	81.6	+0.63
	retrieving@5	83.9	75.1	0.65	0.64	64.3	89.9	+0.80	16.9	68.2	+0.36	48.9	82.8	+0.66
	GARFIELD	**94.6**	**82.6**	**0.83**	**0.72**	**81.1**	**94.6**	**+0.89**	**32.6**	**75.6**	**+0.51**	**65.3**	**88.4**	**+0.77**
Haiku-4.5	closed-book LLM	59.3	53.2	0.42	0.56	36.2	83.4	+0.67	13.5	62.2	+0.24	28.8	76.5	+0.53
	retrieving@3	76.9	69.7	0.54	0.67	51.9	87.6	+0.75	9.0	61.2	+0.22	38.0	79.0	+0.58
	retrieving@5	76.2	67.7	0.52	0.65	52.4	88.4	+0.77	7.9	64.0	+0.28	38.0	80.4	+0.61
	GARFIELD	**94.0**	**76.6**	**0.64**	**0.70**	**71.4**	**92.3**	**+0.85**	**29.2**	**68.5**	**+0.37**	**57.7**	**84.6**	**+0.69**

Table S3: BioProBench results for questions in the *Cell Biology & Culture* domain, with n_{PQA}=316, n_{ERR}=670, and n_{ORD}=363. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result for each LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	75.3	68.8	0.60	0.63	45.4	87.8	+0.76	32.5	74.8	+0.50	36.4	78.6	+0.57
	retrieving@3	83.5	73.0	0.62	0.71	70.4	92.4	+0.85	32.2	75.2	+0.50	43.5	80.3	+0.61
	retrieving@5	85.1	77.3	0.67	0.75	67.6	92.0	+0.84	34.5	75.2	+0.50	44.4	80.2	+0.60
	GARFIELD	**93.0**	**86.9**	**0.91**	**0.82**	**78.7**	**94.6**	**+0.89**	**49.4**	**82.0**	**+0.64**	**58.1**	**85.8**	**+0.72**
GPT-5.4	closed-book LLM	71.5	67.5	0.59	0.60	41.7	86.3	+0.73	32.5	76.0	+0.52	35.3	79.1	+0.58
	retrieving@3	81.6	72.2	0.61	0.70	65.7	92.5	+0.85	32.9	75.6	+0.51	42.7	80.6	+0.61
	retrieving@5	82.6	76.6	0.66	0.74	70.4	93.3	+0.87	29.8	74.6	+0.49	41.9	80.2	+0.60
	GARFIELD	**94.3**	**83.0**	**0.75**	**0.80**	**83.3**	**95.7**	**+0.91**	**48.2**	**81.1**	**+0.62**	**58.7**	**85.5**	**+0.71**
GPT-5.6-sol	closed-book LLM	75.3	72.2	0.64	0.66	50.0	87.9	+0.76	34.9	76.6	+0.53	39.4	80.0	+0.60
	retrieving@3	85.4	73.4	0.63	0.71	69.4	93.7	+0.87	32.9	75.3	+0.51	43.8	80.8	+0.62
	retrieving@5	85.4	76.7	0.67	0.73	68.5	92.9	+0.86	33.7	75.9	+0.52	44.1	81.0	+0.62
	GARFIELD	**97.2**	**88.5**	**0.89**	**0.85**	**83.3**	**95.7**	**+0.91**	**52.2**	**83.9**	**+0.68**	**61.4**	**87.4**	**+0.75**
Sonnet-5	closed-book LLM	71.8	69.6	0.61	0.65	38.9	82.1	+0.64	30.6	72.0	+0.44	33.1	75.0	+0.50
	retrieving@3	81.3	71.5	0.65	0.64	68.5	91.7	+0.83	31.4	71.1	+0.42	42.4	77.2	+0.54
	retrieving@5	82.6	77.5	0.75	0.70	66.7	90.4	+0.81	31.8	71.4	+0.43	42.1	77.1	+0.54
	GARFIELD	**96.5**	**86.9**	**0.89**	**0.82**	**80.6**	**92.6**	**+0.85**	**44.7**	**75.1**	**+0.50**	**55.4**	**80.3**	**+0.61**
Haiku-4.5	closed-book LLM	63.3	53.4	0.46	0.61	33.3	83.4	+0.67	23.9	63.6	+0.27	26.7	69.5	+0.39
	retrieving@3	78.8	66.7	0.55	0.68	60.2	91.0	+0.82	19.2	62.5	+0.25	31.4	71.0	+0.42
	retrieving@5	78.5	69.4	0.57	0.70	57.4	90.7	+0.81	19.6	65.0	+0.30	30.9	72.6	+0.45
	GARFIELD	**92.7**	**82.5**	**0.74**	**0.80**	**76.9**	**94.5**	**+0.89**	**39.2**	**71.1**	**+0.42**	**50.4**	**78.1**	**+0.56**

Table S4: BioProBench results for questions in the *Microbiology & Virology* domain, with n_{PQA}=217, n_{ERR}=101, and n_{ORD}=166. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result in each backbone LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	70.5	67.3	0.65	0.63	48.0	79.5	+0.59	17.1	68.9	+0.38	40.4	76.9	+0.54
	retrieving@3	84.8	73.3	0.67	0.75	68.0	89.3	+0.79	19.5	76.0	+0.52	56.0	**86.0**	**+0.72**
	retrieving@5	85.3	71.3	0.67	0.70	67.2	**89.4**	**+0.79**	17.1	75.2	+0.50	54.8	85.9	+0.72
	GARFIELD	**87.1**	**83.2**	**0.91**	**0.79**	**72.0**	87.1	+0.74	**43.9**	**78.7**	**+0.57**	**65.1**	85.0	+0.70
GPT-5.4	closed-book LLM	71.4	62.4	0.61	0.54	46.4	84.6	+0.69	14.6	72.6	+0.45	38.6	81.6	+0.63
	retrieving@3	83.9	73.3	0.67	0.73	65.6	90.6	+0.81	22.0	77.4	+0.55	54.8	87.4	+0.75
	retrieving@5	80.6	72.3	0.66	0.73	60.8	89.3	+0.79	19.5	76.4	+0.53	50.6	86.1	+0.72
	GARFIELD	**89.9**	**84.2**	**0.79**	**0.84**	**81.6**	**95.0**	**+0.90**	**41.5**	**80.8**	**+0.62**	**71.7**	**91.5**	**+0.83**
GPT-5.6-sol	closed-book LLM	78.8	63.4	0.61	0.57	51.2	84.7	+0.69	26.8	76.0	+0.52	45.2	82.5	+0.65
	retrieving@3	87.6	72.3	0.66	0.73	66.4	91.5	+0.83	22.0	75.5	+0.51	55.4	87.5	+0.75
	retrieving@5	86.6	71.3	0.65	0.72	69.6	91.5	+0.83	19.5	75.5	+0.51	57.2	87.6	+0.75
	GARFIELD	**96.3**	**88.1**	**0.95**	**0.86**	**81.6**	**95.2**	**+0.90**	**43.9**	**83.5**	**+0.67**	**72.3**	**92.3**	**+0.85**
Sonnet-5	closed-book LLM	69.6	63.4	0.59	0.61	42.4	76.4	+0.53	17.1	65.0	+0.30	36.1	73.5	+0.47
	retrieving@3	80.2	73.3	0.73	0.71	64.8	83.8	+0.68	14.6	72.1	+0.44	52.4	80.9	+0.62
	retrieving@5	81.1	72.3	0.71	0.68	64.0	86.2	+0.72	22.0	**73.6**	**+0.47**	53.6	**83.1**	**+0.66**
	GARFIELD	**92.2**	**87.1**	**0.95**	**0.84**	**73.6**	**86.4**	**+0.73**	**31.7**	63.3	+0.27	**63.3**	80.7	+0.61
Haiku-4.5	closed-book LLM	62.2	58.4	0.53	0.63	36.0	82.6	+0.65	9.8	59.2	+0.18	29.5	76.8	+0.54
	retrieving@3	77.9	64.4	0.57	0.69	56.0	85.9	+0.72	12.2	**66.7**	**+0.33**	45.2	81.1	+0.62
	retrieving@5	79.7	69.3	0.61	0.73	57.6	87.2	+0.74	14.6	65.5	+0.31	47.0	81.9	+0.64
	GARFIELD	**92.2**	**86.1**	**0.81**	**0.86**	**72.8**	**91.6**	**+0.83**	**29.3**	63.2	+0.26	**62.0**	**84.6**	**+0.69**

Table S5: BioProBench results for questions in the *Neuroscience Methods* domain, with n_{PQA}=154, n_{ERR}=119, and n_{ORD}=222. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result for each LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	74.0	73.1	0.61	0.64	40.3	84.3	+0.69	30.1	75.6	+0.51	36.9	81.5	+0.63
	retrieving@3	85.1	79.0	0.66	0.74	60.4	90.8	+0.82	30.1	77.4	+0.55	50.5	86.4	+0.73
	retrieving@5	87.0	79.8	0.67	0.75	61.7	90.1	+0.80	38.4	78.9	+0.58	54.1	86.4	+0.73
	GARFIELD	**97.4**	**89.1**	**0.91**	**0.83**	**76.5**	**93.1**	**+0.86**	**52.1**	**84.5**	**+0.69**	**68.5**	**90.3**	**+0.81**
GPT-5.4	closed-book LLM	68.8	73.9	0.63	0.64	35.6	82.2	+0.64	30.1	76.2	+0.52	33.8	80.2	+0.60
	retrieving@3	83.1	81.5	0.73	0.74	61.1	91.4	+0.83	34.2	77.1	+0.54	52.3	86.7	+0.73
	retrieving@5	85.7	80.7	0.70	0.74	60.4	90.1	+0.80	34.2	77.7	+0.55	51.8	86.0	+0.72
	GARFIELD	**95.5**	**87.4**	**0.79**	**0.83**	**74.5**	**93.2**	**+0.86**	**52.1**	**81.7**	**+0.63**	**67.1**	**89.4**	**+0.79**
GPT-5.6-sol	closed-book LLM	76.6	74.8	0.64	0.64	40.9	85.0	+0.70	32.9	77.8	+0.56	38.3	82.6	+0.65
	retrieving@3	87.0	77.3	0.62	0.74	63.1	91.2	+0.82	32.9	77.7	+0.55	53.2	86.7	+0.73
	retrieving@5	86.4	80.7	0.67	0.76	60.4	89.9	+0.80	35.6	78.4	+0.57	52.3	86.1	+0.72
	GARFIELD	**97.4**	**92.4**	**0.95**	**0.89**	**79.2**	**95.9**	**+0.92**	**56.2**	**86.7**	**+0.73**	**71.6**	**92.9**	**+0.86**
Sonnet-5	closed-book LLM	70.8	75.6	0.63	0.68	38.9	81.4	+0.63	28.8	75.1	+0.50	35.6	79.3	+0.59
	retrieving@3	83.1	75.6	0.62	0.69	59.7	87.7	+0.75	31.5	75.0	+0.50	50.5	83.5	+0.67
	retrieving@5	87.0	75.6	0.68	0.66	60.4	88.5	+0.77	34.2	76.6	+0.53	51.8	84.6	+0.69
	GARFIELD	**96.1**	**91.6**	**0.94**	**0.87**	**74.5**	**90.0**	**+0.80**	**45.2**	**77.2**	**+0.54**	**64.9**	**85.8**	**+0.72**
Haiku-4.5	closed-book LLM	60.4	59.7	0.46	0.59	28.9	80.2	+0.60	26.0	62.7	+0.25	27.9	74.5	+0.49
	retrieving@3	80.5	69.7	0.54	0.68	47.7	87.6	+0.75	28.8	65.1	+0.30	41.4	80.2	+0.60
	retrieving@5	78.6	68.1	0.53	0.67	49.0	88.5	+0.77	24.7	65.8	+0.32	41.0	81.1	+0.62
	GARFIELD	**93.5**	**86.6**	**0.80**	**0.81**	**71.1**	**90.5**	**+0.81**	**41.1**	**71.5**	**+0.43**	**61.3**	**84.2**	**+0.68**

Table S6: BioProBench results for questions in the *Bioimaging Technologies* domain, with n_{PQA}=122, n_{ERR}=209, and n_{ORD}=169. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result for each LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	63.9	70.8	0.58	0.66	41.2	85.8	+0.72	21.4	70.9	+0.42	31.4	78.4	+0.57
	retrieving@3	86.9	70.3	0.57	0.68	69.4	92.8	+0.86	22.6	73.8	+0.48	46.2	83.4	+0.67
	retrieving@5	86.1	74.2	0.61	0.71	70.6	93.3	+0.87	21.4	72.3	+0.45	46.2	82.9	+0.66
	GARFIELD	**95.1**	**90.0**	**0.89**	**0.86**	**72.9**	**93.6**	**+0.87**	**32.1**	**77.7**	**+0.55**	**52.7**	**85.7**	**+0.71**
GPT-5.4	closed-book LLM	68.9	66.0	0.53	0.61	37.6	85.1	+0.70	19.0	72.5	+0.45	28.4	78.8	+0.58
	retrieving@3	82.0	71.8	0.59	0.68	65.9	90.0	+0.80	19.0	71.6	+0.43	42.6	80.9	+0.62
	retrieving@5	85.2	73.7	0.60	0.71	62.4	90.0	+0.80	20.2	72.0	+0.44	41.4	81.1	+0.62
	GARFIELD	**95.9**	**82.8**	**0.74**	**0.78**	**72.9**	**93.4**	**+0.87**	**32.1**	**77.1**	**+0.54**	**52.7**	**85.3**	**+0.71**
GPT-5.6-sol	closed-book LLM	72.1	75.6	0.66	0.69	43.5	86.4	+0.73	21.4	73.3	+0.47	32.5	79.9	+0.60
	retrieving@3	86.9	69.9	0.57	0.66	68.2	91.8	+0.84	21.4	73.1	+0.46	45.0	82.5	+0.65
	retrieving@5	87.7	77.0	0.65	0.73	67.1	91.8	+0.84	22.6	72.6	+0.45	45.0	82.3	+0.65
	GARFIELD	**97.5**	**93.3**	**0.91**	**0.91**	**76.5**	**95.4**	**+0.91**	**33.3**	**79.5**	**+0.59**	**55.0**	**87.5**	**+0.75**
Sonnet-5	closed-book LLM	63.1	68.9	0.56	0.64	35.3	82.1	+0.64	21.4	67.7	+0.35	28.4	74.9	+0.50
	retrieving@3	86.9	68.9	0.58	0.61	65.9	89.8	+0.80	20.2	65.9	+0.32	43.2	77.9	+0.56
	retrieving@5	86.1	75.6	0.66	0.68	63.5	**90.4**	**+0.81**	20.2	66.4	+0.33	42.0	78.5	+0.57
	GARFIELD	**96.7**	**90.4**	**0.89**	**0.87**	**69.4**	89.7	+0.79	**29.8**	**71.1**	**+0.42**	**49.7**	**80.5**	**+0.61**
Haiku-4.5	closed-book LLM	59.8	50.2	0.42	0.58	28.2	80.2	+0.60	15.5	60.3	+0.21	21.9	70.3	+0.41
	retrieving@3	86.1	58.4	0.47	0.63	48.2	86.4	+0.73	15.5	63.3	+0.27	32.0	74.9	+0.50
	retrieving@5	82.8	63.6	0.51	0.66	58.8	90.6	+0.81	13.1	61.7	+0.23	36.1	76.2	+0.52
	GARFIELD	**95.9**	**86.6**	**0.78**	**0.83**	**64.7**	**91.4**	**+0.83**	**25.0**	**64.8**	**+0.30**	**45.0**	**78.2**	**+0.56**

Table S7: BioProBench results for questions in the *Biochemical & Molecular Functional Analysis* domain, with n_{PQA}=115, n_{ERR}=132, and n_{ORD}=118. Results are reported for five LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result for each LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	69.6	66.7	0.48	0.54	50.7	87.8	+0.76	26.5	71.9	+0.44	40.7	81.2	+0.62
	retrieving@3	81.7	69.7	0.51	0.64	65.2	91.1	+0.82	30.6	73.8	+0.48	50.8	83.9	+0.68
	retrieving@5	84.3	72.0	0.54	0.64	66.7	92.7	+0.85	28.6	74.2	+0.48	50.8	85.0	+0.70
	GARFIELD	**94.8**	**87.9**	**0.91**	**0.78**	**75.4**	**93.0**	**+0.86**	**40.8**	**79.5**	**+0.59**	**61.0**	**87.4**	**+0.75**
GPT-5.4	closed-book LLM	67.0	62.9	0.44	0.51	47.8	85.8	+0.72	30.6	74.9	+0.50	40.7	81.3	+0.63
	retrieving@3	80.9	68.2	0.50	0.61	63.8	89.9	+0.80	26.5	73.4	+0.47	48.3	83.1	+0.66
	retrieving@5	83.5	74.2	0.56	0.67	62.3	89.5	+0.79	28.6	76.1	+0.52	48.3	83.9	+0.68
	GARFIELD	**93.0**	**84.1**	**0.72**	**0.76**	**84.1**	**94.9**	**+0.90**	**38.8**	**79.8**	**+0.60**	**65.3**	**88.6**	**+0.77**
GPT-5.6-sol	closed-book LLM	76.5	73.5	0.57	0.61	53.6	87.9	+0.76	30.6	74.5	+0.49	44.1	82.3	+0.65
	retrieving@3	80.9	72.0	0.54	0.65	66.7	92.0	+0.84	30.6	74.8	+0.50	51.7	84.8	+0.70
	retrieving@5	82.6	75.0	0.57	0.68	68.1	92.2	+0.84	30.6	77.7	+0.55	52.5	86.2	+0.72
	GARFIELD	**96.5**	**88.6**	**0.80**	**0.83**	**88.4**	**97.0**	**+0.94**	**42.9**	**82.1**	**+0.64**	**69.5**	**90.8**	**+0.82**
Sonnet-5	closed-book LLM	70.4	68.2	0.50	0.60	43.5	84.0	+0.68	**32.7**	70.0	+0.40	39.0	78.2	+0.56
	retrieving@3	80.0	69.7	0.53	0.59	63.8	86.9	+0.74	20.4	67.4	+0.35	45.8	78.8	+0.58
	retrieving@5	78.3	73.5	0.58	0.60	60.9	87.4	+0.75	26.5	**70.4**	**+0.41**	46.6	80.3	+0.61
	GARFIELD	**94.8**	**87.9**	**0.91**	**0.78**	**81.2**	**91.3**	**+0.83**	30.6	68.3	+0.37	**60.2**	**81.8**	**+0.64**
Haiku-4.5	closed-book LLM	64.3	47.0	0.37	0.53	34.8	81.4	+0.63	20.4	55.9	+0.12	28.8	70.8	+0.42
	retrieving@3	72.2	66.7	0.49	0.63	52.2	87.4	+0.75	18.4	62.8	+0.26	38.1	77.2	+0.54
	retrieving@5	73.0	68.9	0.51	0.66	49.3	87.0	+0.74	22.4	**67.1**	**+0.34**	38.1	**78.7**	**+0.57**
	GARFIELD	**91.3**	**84.8**	**0.74**	**0.77**	**69.6**	**88.1**	**+0.76**	**26.5**	57.8	+0.16	**51.7**	75.5	+0.51

Table S8: BioProBench results for questions in the *Immunological Techniques* domain, with n_{PQA}=100, n_{ERR}=192, and n_{ORD}=143. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result within each backbone LLM is shown in **bold**.

		PQA	ERR			ORD								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	72.0	70.3	0.58	0.62	32.1	78.4	+0.57	28.7	74.7	+0.49	30.1	76.2	+0.52
	retrieving@3	84.0	66.7	0.52	0.63	50.0	**87.7**	**+0.75**	27.6	75.4	+0.51	36.4	80.2	+0.60
	retrieving@5	90.0	71.4	0.57	0.67	44.6	85.8	+0.72	28.7	75.6	+0.51	35.0	79.6	+0.59
	GARFIELD	**92.0**	**88.0**	**0.91**	**0.81**	**60.7**	87.4	+0.75	**39.1**	**78.6**	**+0.57**	**47.6**	**82.0**	**+0.64**
GPT-5.4	closed-book LLM	73.0	69.8	0.56	0.60	33.9	80.5	+0.61	27.6	75.1	+0.50	30.1	77.2	+0.54
	retrieving@3	86.0	66.1	0.51	0.64	44.6	83.9	+0.68	26.4	74.1	+0.48	33.6	77.9	+0.56
	retrieving@5	87.0	66.1	0.51	0.63	44.6	83.6	+0.67	26.4	75.1	+0.50	33.6	78.4	+0.57
	GARFIELD	**92.0**	**76.6**	**0.63**	**0.72**	**64.3**	**91.7**	**+0.83**	**42.5**	**80.0**	**+0.60**	**51.0**	**84.6**	**+0.69**
GPT-5.6-sol	closed-book LLM	71.0	72.9	0.62	0.62	41.1	83.1	+0.66	28.7	75.9	+0.52	33.6	78.7	+0.57
	retrieving@3	83.0	63.0	0.49	0.59	46.4	85.1	+0.70	28.7	76.7	+0.53	35.7	80.0	+0.60
	retrieving@5	89.0	67.7	0.53	0.63	44.6	84.4	+0.69	29.9	77.3	+0.55	35.7	80.1	+0.60
	GARFIELD	**98.0**	**89.6**	**0.89**	**0.85**	**64.3**	**91.8**	**+0.84**	**46.0**	**82.7**	**+0.65**	**53.1**	**86.3**	**+0.73**
Sonnet-5	closed-book LLM	76.0	72.4	0.61	0.63	33.9	71.0	+0.42	28.7	67.7	+0.35	30.8	69.0	+0.38
	retrieving@3	88.0	65.1	0.52	0.56	44.6	76.5	+0.53	31.0	**70.4**	**+0.41**	36.4	72.8	+0.46
	retrieving@5	87.0	75.0	0.64	0.65	48.2	77.6	+0.55	27.6	67.3	+0.35	35.7	71.3	+0.43
	GARFIELD	**95.0**	**86.5**	**0.89**	**0.79**	**60.7**	**83.6**	**+0.67**	**35.6**	68.5	+0.37	**45.5**	**74.4**	**+0.49**
Haiku-4.5	closed-book LLM	60.0	51.6	0.42	0.58	26.8	75.0	+0.50	19.5	58.5	+0.17	22.4	65.0	+0.30
	retrieving@3	79.0	60.9	0.47	0.61	33.9	78.2	+0.56	16.1	56.4	+0.13	23.1	65.0	+0.30
	retrieving@5	84.0	59.9	0.47	0.61	35.7	79.5	+0.59	18.4	**64.0**	**+0.28**	25.2	70.1	+0.40
	GARFIELD	**99.0**	**74.0**	**0.59**	**0.70**	**57.1**	**88.0**	**+0.76**	**31.0**	61.9	+0.24	**41.3**	**72.1**	**+0.44**

Table S9: BioProBench results for questions in the *Genomics Technologies* domain, with n_{PQA}=335, n_{ERR}=223, and n_{ORD}=300. Results are reported across five backbone LLMs under four settings, with ORD further separated into child-step, top-step, and overall ordering. The best result within each backbone LLM is shown in **bold**.

		PQA	**ERR**			**ORD**								
						child			*top*			*ALL*		
Model	**Setting**	Acc.(%)	Acc.(%)	Prec.	F1	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ	Seq-EM(%)	PW(%)	τ
Opus-4.8	closed-book LLM	69.6	70.0	0.60	0.64	49.2	86.2	+0.72	20.4	74.2	+0.48	38.3	81.7	+0.63
	retrieving@3	83.6	78.9	0.67	0.77	67.9	**93.4**	**+0.87**	23.9	76.2	+0.52	51.3	86.9	+0.74
	retrieving@5	86.0	81.6	0.71	0.79	67.4	93.1	+0.86	20.4	74.1	+0.48	49.7	86.0	+0.72
	GARFIELD	**93.7**	**93.3**	**1.00**	**0.90**	**70.6**	90.9	+0.82	**38.1**	**80.5**	**+0.61**	**58.3**	**87.0**	**+0.74**
GPT-5.4	closed-book LLM	71.9	72.6	0.66	0.63	46.5	86.6	+0.73	17.7	72.2	+0.44	35.7	81.2	+0.62
	retrieving@3	82.7	78.5	0.68	0.75	64.7	**92.4**	**+0.85**	21.2	74.5	+0.49	48.3	85.6	+0.71
	retrieving@5	85.4	81.6	0.72	0.78	63.6	91.9	+0.84	16.8	73.8	+0.48	46.0	85.1	+0.70
	GARFIELD	**93.4**	**88.3**	**0.82**	**0.86**	**72.7**	91.7	+0.83	**38.9**	**81.1**	**+0.62**	**60.0**	**87.7**	**+0.75**
GPT-5.6-sol	closed-book LLM	74.3	75.3	0.68	0.68	52.9	89.0	+0.78	20.4	73.7	+0.47	40.7	83.2	+0.66
	retrieving@3	87.8	81.6	0.71	0.79	70.1	92.9	+0.86	20.4	74.0	+0.48	51.3	85.8	+0.72
	retrieving@5	89.9	85.2	0.76	0.82	70.1	92.9	+0.86	23.0	75.1	+0.50	52.3	86.2	+0.72
	GARFIELD	**97.9**	**94.2**	**0.94**	**0.92**	**77.0**	**93.6**	**+0.87**	**44.2**	**83.7**	**+0.67**	**64.7**	**89.8**	**+0.80**
Sonnet-5	closed-book LLM	70.1	72.2	0.63	0.66	44.4	80.4	+0.61	17.7	68.8	+0.38	34.3	76.0	+0.52
	retrieving@3	85.7	80.3	0.73	0.75	65.2	88.5	+0.77	22.1	69.8	+0.40	49.0	81.5	+0.63
	retrieving@5	85.1	80.7	0.77	0.74	62.6	88.3	+0.77	17.7	68.9	+0.38	45.7	81.0	+0.62
	GARFIELD	**95.8**	**95.1**	**1.00**	**0.93**	**72.2**	**89.5**	**+0.79**	**38.9**	**77.6**	**+0.55**	**59.7**	**85.0**	**+0.70**
Haiku-4.5	closed-book LLM	61.2	52.5	0.44	0.58	35.3	82.7	+0.65	14.2	67.8	+0.36	27.3	77.1	+0.54
	retrieving@3	76.7	70.0	0.57	0.71	53.5	88.7	+0.77	13.3	67.1	+0.34	38.3	80.6	+0.61
	retrieving@5	78.2	70.4	0.57	0.71	54.0	88.7	+0.77	12.4	65.9	+0.32	38.3	80.1	+0.60
	GARFIELD	**93.7**	**85.2**	**0.76**	**0.83**	**67.4**	**90.7**	**+0.81**	**33.6**	**75.8**	**+0.52**	**54.7**	**85.1**	**+0.70**

C Open-Ended Protocol Question Answering by Reasoning Depth

Tables S11–S13 provide the full BioPIE results separately for single-, double-, and triple+-hop questions. Across all three reasoning depths, GARFIELD achieves the highest EM, Token-F1, and judge scores for all five backbone LLMs. Averaged across the five LLMs, GARFIELD obtains EM scores of 58.88%, 53.96%, and 53.44% for single-, double-, and triple+-hop questions, respectively, compared with 25.86%, 23.72%, and 24.72% for retrieving@5. The LLM-as-a-judge evaluations show the same trend: averaging J_O and J_G across the five backbone LLMs, GARFIELD achieves 83.22%, 77.82%, and 74.54% across the three reasoning depths, compared with 57.11%, 53.44%, and 55.82% for retrieving@5. This corresponds to judge-score gains of 26.11%, 24.38%, and 18.72% for single-, double-, and triple+-hop questions, respectively. Together, the EM, Token-F1, and LLM-as-a-judge evaluations show that GARFIELD consistently outperforms the retrieval-based baselines within each reasoning-depth subset, with substantial gains observed for single-, double-, and triple+-hop questions. Also, to validate our LLM-as-judge BioPIE evaluation pipeline, we sampled 200 questions spanning five models, four settings, and three hop groups. Two PhD students with backgrounds in computer science, electrical, and biomedical engineering independently judged each answer's correctness. We compared the human labels with the automated judges (Table S10). Strong annotator-judge and inter-annotator agreement supports the reliability of the LLM-as-judge automated evaluation.

Table S10: Agreement between human annotators and LLM judges on 200 BioPIE questions. Accuracy and Cohen's κ indicate strong alignment between human judgments and the LLM-as-judge evaluation.

	Opus-4.8		GPT-5.4	
Metric	Acc. (%)	Cohen's κ	Acc. (%)	Cohen's κ
Annotator 1 vs. LLM Judge	93.0	0.856	94.5	0.886
Annotator 2 vs. LLM Judge	94.0	0.875	95.5	0.906
Annotator 1 vs. Annotator 2	96.0	0.918	96.0	0.918

Table S11: BioPIE open-ended question answering results on the 2,935 single-hop questions across five backbone LLMs under four settings. J_O and J_G use Opus-4.8 and GPT-5.4 as judges, respectively. The best result within each backbone LLM is shown in **bold**.

Model	Setting	EM (%)	Token-F1	J_O (%)	J_G (%)
Opus-4.8	closed-book LLM	12.8	0.280	36.6	39.4
	retrieving@3	30.5	0.480	58.1	60.8
	retrieving@5	36.4	0.494	59.5	60.6
	GARFIELD	**58.9**	**0.746**	**83.9**	**82.8**
GPT-5.4	closed-book LLM	10.9	0.265	32.2	36.4
	retrieving@3	21.0	0.440	56.4	59.1
	retrieving@5	29.4	0.470	58.5	60.9
	GARFIELD	**60.1**	**0.753**	**84.2**	**82.5**
GPT-5.6-sol	closed-book LLM	11.8	0.281	36.0	39.1
	retrieving@3	13.2	0.374	58.7	61.5
	retrieving@5	18.6	0.400	60.9	62.8
	GARFIELD	**59.7**	**0.759**	**86.2**	**85.1**
Sonnet-5	closed-book LLM	9.9	0.249	34.0	36.3
	retrieving@3	29.3	0.458	56.4	58.7
	retrieving@5	30.5	0.451	56.6	57.9
	GARFIELD	**58.4**	**0.746**	**84.7**	**83.5**
Haiku-4.5	closed-book LLM	7.9	0.184	26.3	29.2
	retrieving@3	12.9	0.248	47.2	48.0
	retrieving@5	14.4	0.258	45.9	47.5
	GARFIELD	**57.3**	**0.712**	**80.1**	**79.2**

Table S12: BioPIE open-ended question answering results on the 285 double-hop questions across five backbone LLMs under four settings. J_O and J_G use Opus-4.8 and GPT-5.4 as judges, respectively. The best result within each backbone LLM is shown in **bold**.

Model	Setting	EM (%)	Token-F1	J_O (%)	J_G (%)
Opus-4.8	closed-book LLM	13.0	0.268	33.7	37.5
	retrieving@3	27.4	0.429	52.6	56.5
	retrieving@5	31.2	0.439	54.7	56.5
	GARFIELD	**52.3**	**0.675**	**75.4**	**74.7**
GPT-5.4	closed-book LLM	9.1	0.266	33.7	38.2
	retrieving@3	21.8	0.439	55.8	60.0
	retrieving@5	30.2	0.467	55.4	58.6
	GARFIELD	**57.5**	**0.719**	**80.4**	**79.6**
GPT-5.6-sol	closed-book LLM	13.7	0.283	34.7	39.3
	retrieving@3	14.0	0.356	56.1	60.4
	retrieving@5	18.6	0.391	59.3	61.1
	GARFIELD	**54.4**	**0.704**	**80.7**	**80.0**
Sonnet-5	closed-book LLM	12.6	0.269	35.1	35.8
	retrieving@3	26.0	0.410	51.2	54.0
	retrieving@5	26.3	0.401	49.5	52.3
	GARFIELD	**57.9**	**0.732**	**82.5**	**82.1**
Haiku-4.5	closed-book LLM	7.7	0.180	26.7	28.8
	retrieving@3	10.5	0.212	42.8	46.0
	retrieving@5	12.3	0.215	41.4	45.6
	GARFIELD	**47.7**	**0.631**	**71.9**	**70.9**

Table S13: BioPIE open-ended question answering results on the 55 triple+-hop questions across five backbone LLMs under four settings. J_O and J_G use Opus-4.8 and GPT-5.4 as judges, respectively. The best result within each backbone LLM is shown in **bold**.

Model	Setting	EM (%)	Token-F1	J_O (%)	J_G (%)
Opus-4.8	closed-book LLM	16.4	0.294	30.9	34.5
	retrieving@3	27.3	0.463	60.0	58.2
	retrieving@5	38.2	0.498	56.4	56.4
	GARFIELD	**52.7**	**0.693**	**74.5**	**74.5**
GPT-5.4	closed-book LLM	12.7	0.274	29.1	30.9
	retrieving@3	16.4	0.431	60.0	63.6
	retrieving@5	21.8	0.435	60.0	60.0
	GARFIELD	**54.5**	**0.687**	**76.4**	**74.5**
GPT-5.6-sol	closed-book LLM	16.4	0.302	27.3	32.7
	retrieving@3	12.7	0.374	60.0	60.0
	retrieving@5	23.6	0.408	60.0	58.2
	GARFIELD	**61.8**	**0.778**	**85.5**	**83.6**
Sonnet-5	closed-book LLM	16.4	0.295	32.7	32.7
	retrieving@3	20.0	0.419	54.5	52.7
	retrieving@5	27.3	0.471	60.0	61.8
	GARFIELD	**49.1**	**0.661**	**70.9**	**70.9**
Haiku-4.5	closed-book LLM	9.1	0.193	18.2	23.6
	retrieving@3	12.7	0.240	40.0	41.8
	retrieving@5	12.7	0.234	41.8	43.6
	GARFIELD	**49.1**	**0.614**	**67.3**	**67.3**

D TOOL-CALL DISTRIBUTIONS

As shown in Figure S2 and Figure S1, tool-call distributions are strongly right-skewed, with most queries requiring only a few calls and a smaller subset producing longer trajectories. Across BioPIE question runs, the median was 4 calls (interquartile range, IQR: 2-8), with 75.1% requiring at most 8 calls. Tool use was higher for questions requiring deeper reasoning, with the mean increasing from 7.88 calls for single-hop questions to 9.55 calls for triple+-hop questions. Across the 30,000 BioProBench question runs, tool-call trajectories were shorter overall, with a median of 2 calls (IQR: 1-2), and 89.9% of runs requiring at most 4 calls. Tool use also differed by task, with PQA and ERR requiring more calls on average (3.08 and 2.82, respectively) than ORD (1.59). Finally, trajectory length varied substantially across backbone LLMs, with Claude Haiku 4.5 producing the longest trajectories on both BioPIE and BioProBench, averaging 16.44 and 3.82 calls per query, respectively.

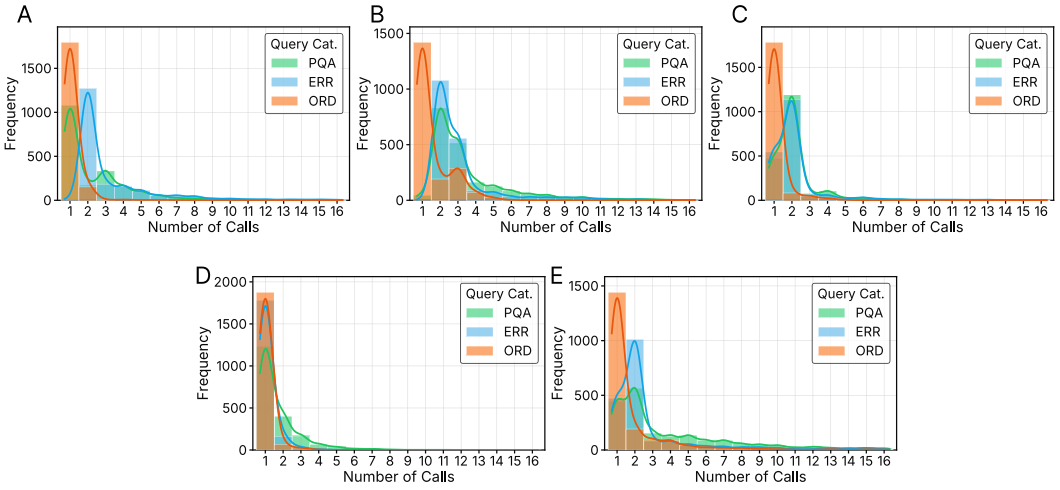

Figure S1: Distribution of tool calls for BioProBench question run, shown separately for PQA, ERR, and ORD questions across five backbone LLMs: A) GPT-5.6 Sol, B) GPT-5.4, C) Sonnet 5, D) Opus 4.8, and E) Haiku 4.5.

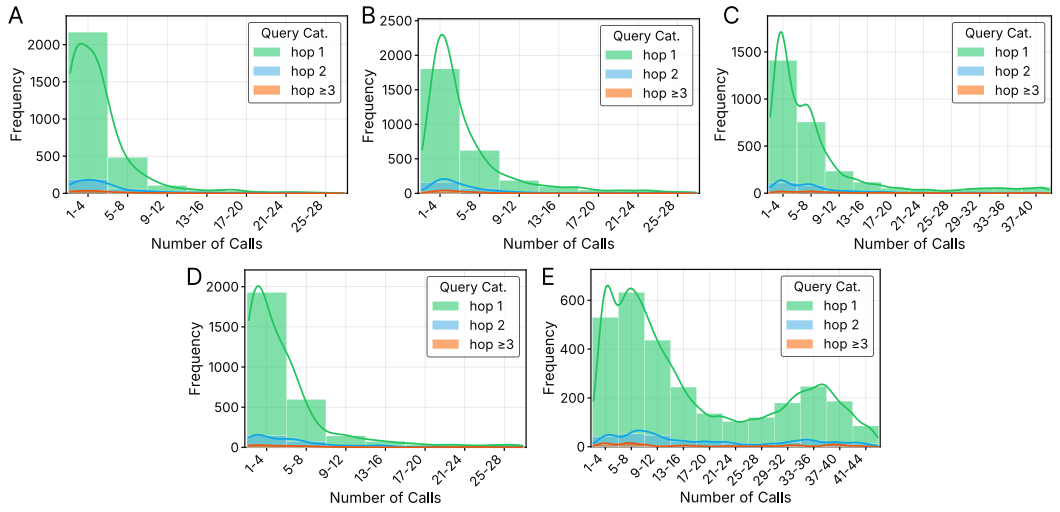

Figure S2: Distribution of tool calls for BioPIE question run, shown separately for single-, double-, and triple+-hop questions across five backbone LLMs: A) GPT-5.6 Sol, B) GPT-5.4, C) Sonnet 5, D) Opus 4.8, and E) Haiku 4.5.

E PROMPTS FOR MULTI-VIEW EXPERIMENTAL PROTOCOL GRAPH CONSTRUCTION

We include here the complete prompts used to construct the Multi-View Experimental Protocol Graph Representation (Section 3.1) for reproducibility. Figure S3 shows the Hierarchical View Extractor Prompt used to recover the Protocol → Process → Step → Action structure. Figure S4 shows the Preprocessor Prompt used to resolve anaphoric references and expand abbreviations before downstream extraction. Figures S5 and S6 show the prompts used to extract execution structure and Action Context information, respectively. Figure S7 shows the Entity Normalization Prompt used to map local Actions and experimental objects to canonical representations for cross-protocol alignment.

Hierarchical View Extractor Prompt

You are the HIERARCHICAL VIEW Extractor, responsible for recovering the hierarchical organization of an experimental protocol:

Protocol → Process → Step → Atomic Action

You perform BOTH high-level segmentation into Processes and Steps and low-level segmentation of each Step into Atomic Actions. The protocol is provided as numbered source lines in canonical order; preserve this order. Lines marked [heading] are usually section titles but may also contain executable instructions.

The Hierarchical View contains four levels:

Protocol. The complete laboratory procedure, including all operations, reagents, instruments, conditions, and quality controls required to produce the intended experimental outcome. The Protocol is given as input and forms the root of the hierarchy. *Example: RNA extraction protocol.*

Process. A major stage of the Protocol formed by a "contiguous" sequence of Steps that together accomplish one intermediate experimental objective. *Example: Cell lysis.*

Step. One executable instruction performed before proceeding to the next. A Step usually corresponds to one source line, but consecutive lines may be merged when they express one instruction, and one line may be split when it contains multiple independent instructions. *Example: Add lysis buffer and incubate for 10 min.*

Atomic Action. One indivisible laboratory operation within a Step, defined by a single operative verb together with its arguments, such as the material, amount, duration, temperature, speed, or location. A Step may contain multiple Atomic Actions. *Example: Add lysis buffer.*

Operational Principle

Construct Processes and Steps by referencing source line numbers and preserve source order. All source text must remain VERBATIM. The only authored text is (1) a Process title when no suitable source heading exists and (2) the surface verb of an Atomic Action.

Process titles. Use `{"title_src": N}` when source line N is a [heading] that names the Process. The line is consumed as the title and must not appear again as a Step. Use `{"title": "<your text>"}` only when no suitable [heading] exists or when the heading contains executable instructions. Authored titles must be concise and technically specific (3–8 words).

Steps. Represent each Step in one of three forms: **Single line:** `{"src": [N], "actions": [...]}` when one source line expresses one Step; **Merge:** `{"src": [N, N+1, ...], "actions": [...]}` when consecutive lines together express one instruction; or **Split:** `{"split": N, "parts": ["<verbatim substring>", ...], "actions_per_part": [[...], ...]}` when one line contains multiple independent instructions. Split parts must be copied VERBATIM from source line N and remain in source order.

Atomic actions. Represent each Atomic Action as `{"verb": "<action verb>", "text": "<verbatim action text>"}`. The **verb** is a 1-3 word experimental operation verb such as add, incubate, centrifuge, wash, transfer, mix, or measure. The **text** is the exact substring of the Step expressing that action. Multiple actions must use disjoint substrings in source order; short connectives such as "then", "and", or "next" may be omitted.

Non-executable text. If a Step contains no executable operation, such as a section label or purely descriptive statement, return `{"actions": []}`. Never fabricate an Atomic Action.

Mandatory Rules

Preserve source order. Emit all source lines in their original order. Never reorder instructions based on expected laboratory workflow.

Cover every line exactly once. Every source line must appear exactly once as a Step through `src` or `split`, or as a Process title through `title_src`. This includes [heading] lines.

Merge and split correctly. Merge only adjacent source lines that together form one instruction. Split only within a single source line, using VERBATIM substrings of that line.

Keep Processes contiguous. Each Process must cover a contiguous run of source lines. Use the run's [heading] as `title_src` when available; author a title only when no suitable heading exists or the heading is itself executable.

Treat executable headings as Steps. A [heading] that contains an operative verb and executable information, such as an amount, time, temperature, or speed, is a Step rather than a Process title. Preserve and decompose its executable content into Atomic Actions.

Do not author Step text. Step text must come directly from the source. The only authored text is a Process title when necessary.

Preserve Atomic Action granularity. Separate distinct operations into different Atomic Actions, e.g., "do X, then do Y." Keep one operation with multiple parameters as a single Atomic Action, e.g., "Centrifuge at 3000 g for 5 min at 4 C."

Examples

{Example} + {Correct Answer} + {Incorrect Answer} + {Notes}

Output Schema

{Schema}

Input Protocol Title and Content

Protocol title: {title}

{Protocol content}

Figure S3: Hierarchical View Extractor Prompt used to decompose experimental protocols into Protocol, Process, Step, and Atomic Action levels.

The Preprocessor

You are the Preprocessor. Your responsibility is to prepare the ATOMIC ACTION texts of a laboratory protocol for downstream structured extraction through two local lexical substitutions: (1) Anaphor resolution and (2) Abbreviation expansion.

Operational principle

(1) **Anaphor resolution.** Resolve an anaphoric reference only when its antecedent is unambiguous from earlier protocol context.

Entity references. Resolve generic references to previously introduced samples, mixtures, solutions, products, or fractions, such as "the sample", "the solution", "the pellet", or pronouns such as "it" and "this".

Equipment references. Resolve references such as "the tube", "the plate", or "the column" when exactly one previously introduced piece of equipment matches.

Parameter references. Resolve references such as "the volume", "the concentration", or "the temperature" when the attribute clearly belongs to one previously introduced entity.

Resolution rule. Search prior Processes, Steps, and Actions in reverse order and select the most specific valid antecedent. If multiple antecedents are plausible, leave the reference unchanged.

(2) **Abbreviation expansion.** Expand every confidently identifiable abbreviation, chemical formula, or technical shorthand using the format **full form (abbreviation)**. Expand every occurrence.

```
"PBS"  -> "phosphate-buffered saline (PBS)"
"NaCl" -> "sodium chloride (NaCl)"
"PCR"  -> "polymerase chain reaction (PCR)"
```

What to expand. Expand chemical formulas, biological shorthand, named techniques, reagent or buffer abbreviations, and instrument abbreviations with a well-defined full form.

Ambiguous abbreviations. When an abbreviation has multiple meanings, use the expansion supported by the protocol context. If the protocol explicitly defines it, use that definition consistently.

Mandatory Rules

One output per Atomic Action. Emit one rewritten text for each Atomic Action, keyed by `(step_index, action_index)`.

Use protocol-level context. Resolve anaphors using the full segmented protocol, including prior Processes, Steps, and Actions. An antecedent may occur several actions earlier or in a Process title. Do not resolve an anaphor from the current action text alone.

Rewrite minimally. Change only the text required to resolve an anaphor or expand an abbreviation. If neither applies, return the Atomic Action text VERBATIM.

Preserve the original text. Do not paraphrase, shorten, reorder, or normalize wording or verbs. Do not change numbers, quantities, units, capitalization, or punctuation. Do not correct typos or introduce information not present in the protocol.

Examples

{Example} + {Correct Answer} + {Incorrect Answer} + {Notes}

Output Schema

{Schema}

Input Protocol Title and Content

Protocol title: {title}

{Segmented protocol content}

Figure S4: Preprocessor Prompt used to resolve anaphoric references and expand abbreviations in Atomic Action texts before downstream graph extraction.

Execution View Extractor Prompt

You are the Execution View Extractor. Your task is to recover the execution structures of a laboratory protocol: repeated execution, condition-dependent execution, and parallel execution.

Operational principle

For each execution structure, identify the element it acts on and provide a VERBATIM 3–12 word evidence span. Record the elements affected by the structure rather than the instruction that names it. Never infer a structure that is not explicitly stated in the protocol.

Execution structures

(1) **Repeated execution.** A run of Processes, Steps, or Atomic Actions that the text explicitly declares to repeat. Record any stated repetition count or termination condition.

Protocol examples.

```
"Repeat steps 4-5 twice"
"repeat until the pellet is white"
```

(2) **Condition-dependent execution.** A branching point where one or more Steps execute according to a condition, preference, or choice. Each affected Step must belong to exactly one outcome category: `if_true`, `if_false`, `optional`, or `alternative`.

Verifiable condition. Use `if_true` when execution depends on a condition that can be objectively checked at that point in the protocol, and `if_false` for an explicitly stated else branch.

Optional execution. Use `optional` when execution depends on preference, availability, or convenience rather than an objectively verifiable condition.

Alternative execution. Use `alternative` for mutually exclusive choices between two or more execution paths.

Protocol examples.

```
"If the supernatant is turbid, filter it" -> filter step in if_true
"Optionally, store the sample"            -> store step in optional
```

(3) **Parallel execution.** Two or more Processes, Steps, or Atomic Actions that the text explicitly states execute at the same time rather than one after another.

Protocol examples.

```
"Run the PCR while preparing the gel"
"During the incubation, prepare the wash buffer"
```

Hierarchy awareness

Represent repeated and parallel execution at the hierarchy level explicitly indicated by the source text.

Process level. Reference Process indices when the structure applies to an entire Process, such as "repeat the entire purification phase twice".

Step level. Reference Step indices when the structure applies to complete Steps, such as "repeat steps 4-5 twice".

Atomic Action level. Reference `(step, action)` index pairs when the structure applies to individual Atomic Actions within a Step.

Decision targets. Decisions always route to Steps (Decision → Step), so all decision outcomes use Step indices.

Examples

{Example} + {Correct Answer} + {Incorrect Answer} + {Notes}

Output Schema

{Schema}

Input Protocol Title and Content

Protocol title: {title}

{Segmented protocol content}

Figure S5: Execution View Extractor Prompt used to identify explicitly stated repeated, condition-dependent, and parallel execution structures and their supporting evidence in experimental protocols.

Action Context View Extractor Prompt

You are the ACTION CONTEXT VIEW Extractor. Your responsibility is to extract the structured context associated with each ATOMIC ACTION of a laboratory protocol, including materials, biological samples, instruments, outputs, parameters, locations, methods, modifiers, required conditions, and rationale. Together, these per-action records form the ACTION CONTEXT VIEW of the protocol.

The Atomic Actions have already been prepared by upstream components:

- The Hierarchical View Extractor segmented the protocol into Processes → Steps → Atomic Actions and assigned each Atomic Action a verb tag and verbatim action text and the Preprocessor resolved anaphoric references and expanded abbreviations in the action text.

Do NOT re-segment, merge, split, reorder, drop, or invent actions. For each provided Atomic Action, emit exactly one enriched record containing only the contextual information stated in that action's text.

Operational principle

(1) Extraction principles

- Extract ONLY what the action text states. Never invent facts, quantities, or entities.

- BE THOROUGH. Prefer over-extraction of stated details to under-extraction. Do not filter details on the grounds of triviality; if the action names it, capture it.

- Do not restate the action verb as a purpose or rationale.

- Do not repeat the same phrase under more than one field.

- Per-field "evidence" is a short EXACT quote (3–12 consecutive words) copied verbatim from that action's text, so the field's grounding can be located by string search. Distinct records within a field get distinct quotes.

(2) Object identity and coreference

Tag every material / biological_sample / instrument / output mention with a "ref" id. Mint ids separately by kind:

materials m1, m2, ... | biological_samples s1, s2, ... | instruments i1, i2, ... | outputs o1, o2, ...

- SAME object, different words → reuse the existing ref. This includes definite or pronoun mentions that unambiguously refer to an earlier object.

- SAME words, different object → use different refs.

- "name" is the EXACT surface used in that action's text. Express identity through "ref". (Outputs are the one exception, see the outputs section.)

PER-ACTION FIELDS

Biological samples `{"name","ref"}`. Biological specimens or specimen-derived preparations originating from cells, organisms, or biological material. This includes cells and cultures, tissues, whole organisms, bodily fluids, subcellular fractions, and isolated biomolecular preparations such as DNA, RNA, proteins, lysates, and immunoprecipitates.

Materials `{"name","ref"}`. Substances and supplies used in the protocol, including reagents, chemicals, buffers, media, kits, gels, salts, solvents, enzymes, and antibodies.

Instruments `{"name","ref"}`. A specific piece of equipment, device, software tool, or container used in the action.

(1) Generic references such as "the device" or "the system" that refer to previously identified equipment reuse the existing instrument ref.

(2) A part or sub-region of an instrument, such as the bottom of a tube, wells of a plate, or wall of a flask, is NOT a separate instrument. Represent it as a LOCATION with `of` set to the parent instrument ref.

Outputs `{"name","ref"}`. What the action PRODUCES, such as a mixture, solution, pellet, supernatant, eluate, lysate, reaction product, filtrate, or purified fraction. Assign a concise technical name that best describes the output; the name does not need to be verbatim from the action text.

parameters {"type","value","unit","of","relative","reference","evidence"}. "of" = the ref id (or name) of the object a value characterises when it is not the action itself, else null. CAPTURE EVERY numeric value the action states: volumes, concentrations, amounts (mass or moles), temperatures, times, speeds (g or rpm), pH, wavelengths, counts, cycles. For a RECIPE or TABLE action (a buffer or master-mix recipe, a thermocycler program, an instrument-settings block), emit a SEPARATE parameter for EACH component's amount AND each final concentration, with "of" set to that component's ref. Never capture the component names while dropping their amounts. OMIT a parameter entirely if it has no value (never emit a null or empty value).

Recipe When an action defines a mixture, buffer, master mix, or reaction from multiple components, extract the product as an output and each component as a material. Associate each stated amount and final concentration with the corresponding material through its `ref`. Use available table headers or labels to preserve the correct component-column alignment, including when the table has been flattened into text. Preserve stock labels in material names, capture the stated total volume, and do not infer missing values.

Example. "Make 10 mL Buffer X: 1 M NaCl 1 mL (100 mM); glycerol 2 mL (20%); water 7 mL" yields output "Buffer X", materials "1 M NaCl", "glycerol", and "water", with their corresponding volume and concentration parameters.

Locations WHERE, spatially. Two shapes, both {"name","ref","of","evidence"}: (a) SITE of the action: where the operation itself takes place ("on ice", "in the fume hood", "on the bench"). Set "of" = null. (b) LOCATION of an object: a spatial part / sub-region of an already-named instrument where a material/sample is placed ("each well", "the bottom of the tube", "well A1", "the top chamber"). Set "of" = the ref of the parent object (usually an instrument, sometimes a sample). Reuse the same "ref" across actions when the same location recurs (mint per protocol: l1, l2, ...). "name" is the exact surface phrase.

Methods NAMED TECHNIQUES / ASSAYS explicitly invoked ("PCR", "Western blot", "gel electrophoresis", "immunoprecipitation", "flow cytometry"). Emit ONLY when the action names a recognizable technique, not for ordinary verbs like centrifuge / wash / mix. {"name","evidence"}

Modifiers manner qualifiers – HOW the action is performed. Short adverbs ("gently", "quickly", "carefully", "vigorously", "briefly", "dropwise"). NOT a technique (that is methods), NOT a numeric parameter, NOT a purpose. {"text","evidence"}

Required_conditions qualitative, NON-NUMERIC preconditions or constraints that MUST hold around the action – properties of the environment, safety requirements, the workspace, or the reagents that are asserted as necessary for success. Think "checkbox" facts a lab tech must satisfy before or while performing the action. NOT a numeric parameter, NOT a spatial location, NOT a manner adverb, NOT a WHY explanation. {"text","evidence"}

EXAMPLES

```
"Under sterile conditions, add..."      -> "sterile conditions"
"Perform in a laminar flow hood"        -> "in a laminar flow hood"
```

Rationale: the WHY – any stated reason, intent, goal, or explanation. A goal ("to pellet the cells") and a cause ("because the protein is fragile") both live here. HARD RULE: rationale requires an EXPLICIT reason/goal marker ("to ...", "so that ...", "in order to ...", "because ...", "since ...", "as ...", "due to ..."). Do NOT infer or invent one. Do NOT include: temporal markers ("before use", "after that", "then", "next"), the action verb itself, a bare noun phrase with no reason marker, anything that is really a location, parameter, modifier, or required_condition. {"text","evidence"}

Examples

{Example} + {Correct Answer} + {Incorrect Answer} + {Notes}

Output Schema

{Schema}

Input Protocol Title and Content

Protocol title: {title}

{Segmented protocol content}

Figure S6: Action Context View Extractor Prompt used to extract the experimental objects and contextual information associated with each Atomic Action, including parameters, locations, methods, conditions, rationale, and recipes.

Entity Normalization Prompt

You normalize {ENTITY_KIND} from experimental protocols to a standardized canonical name, in the style of experimental ontologies.

Mandatory Rules

- Remove quantities, concentrations, volumes, sizes, temperatures, colors, and handling adjectives ("cold", "new", "fresh", "sterile", "1 M", "70%"). Keep a size or format only when it is part of the standard product name.

- Lowercase chemical names; standard capitalization for brand/product names.

- Also output 'parent': the immediate generic CATEGORY the entity belongs to, as a short ontology-style noun phrase. The parent MUST be strictly broader than the canonical name. NEVER identical or near-identical to it. Always give a non-empty parent; if truly unsure, fall back to a broad class ("reagent", "solution", "labware", "consumable", "instrument", "equipment", "software", "operation", "sample", "product").

Examples

{Example} + {Correct Answer} + {Incorrect Answer} + {Notes}

Output Schema

{Schema}

Input

```
1. label={LABEL} | mention="<surface_name>" | evidence="<one sentence, <=200 chars>"
2. label={LABEL} | mention="<surface_name>" | evidence="<one sentence, <=200 chars>"
... (batch of up to PASS1_BATCH items)
```

Figure S7: Entity Normalization Prompt used to map local Actions and experimental objects to canonical names and broader parent categories for cross-protocol alignment.

F GRAPH-AWARE AGENT: PROMPT, SCHEMA, AND TOOL INTERFACE

This section provides the complete components used to instantiate the graph-aware GARFIELD agent described in Section 3.2. Figure S8 presents the full agent prompt, which specifies the iterative reasoning and tool-use procedure, including schema-guided tool selection, accumulation of retrieved evidence in Memory, dynamic tool discovery through TOOLRAG, and generation of the final answer from graph-grounded evidence. Figure S9 provides the Graph Schema Card supplied to the agent, describing the entities and relations available across the hierarchical, execution, action-context, and cross-protocol alignment views of the Multi-View Experimental Protocol Graph Representation. Table S14 lists the graph tools available to the agent and summarizes the graph operation supported by each tool.

Graph-Aware GARFIELD Agent Prompt

You are the GARFIELD agent, a graph-augmented reasoning agent that works over a multi-view experimental protocol graph representation. Your responsibility is to determine what information is required to answer the query, retrieve that information using the available graph tools, and iteratively use the returned observations to guide next reasoning steps and tool selection. Do not rely on unsupported assumptions when the required information can be obtained from the graph.

GRAPH SCHEMA

The following schema card describes the entity types, relation types, and connectivity represented in the multi-view experimental protocol graph representation:

{schema_card}

Use the schema to understand:

• what information is represented in the graph and how different parts of the graph are connected;

• which entities and relations are relevant to the query;

• which graph views may contain the required evidence.

Use the graph schema and its entity and relation structure to guide reasoning and tool selection.

AVAILABLE TOOLS

The following tools are available at the current step:

{tools}

Each tool provides access to a specific part, relation, or view of the multi-view experimental protocol graph representation.

Select tools based on the information required to answer the query.

A special tool, ToolRAG, can retrieve additional tools from the full graph-tool library when the currently available tools do not meet the required needs.

WORKFLOW

At each step:

1. Examine the user query, the graph schema, the Memory from previous steps, and the currently available tools.

2. Determine what information is still required to answer the query.

3. Generate a concise reasoning summary describing:

- what is currently known from the retrieved evidence; and
- what information should be obtained next.

4. Select one or more available tools that can retrieve the required information.
5. When multiple independent pieces of evidence are required and the relevant tools are already available, they may be called together in the same step.
6. Execute the selected tools and examine the returned observations.
7. Use the new observations together with the accumulated Memory to determine whether:

- additional information is required;
- another graph view or relation should be explored; or
- the available evidence is sufficient to answer the query.

8. If additional information is required, continue with another reasoning and tool-use step.
9. If the required capability is not available in the current tool set, use ToolRAG to retrieve appropriate tools.
10. When sufficient evidence has been collected, return the final answer.

TOOL SELECTION

Use the graph schema to identify relevant entity types, relations, and connections before selecting a tool. Choose tools according to the information needed to resolve the query.
If answering the query requires information from different graph views, entity types, or relations, use the relevant tools together and combine their returned evidence.
Do not repeat a tool call when its result has already provided the required information unless additional retrieval is necessary.

TOOLRAG

Use ToolRAG only when the currently available tools cannot retrieve the information from the graph that the agent determines is needed to answer the query.
When calling ToolRAG, describe the required capability in natural language. Do not guess the name of a tool.
Example:

```
{"tool": "ToolRAG", "args": {"need": ["retrieve the operations that occur before and after a given experimental action"], "k": 6}}
```

The tools returned by ToolRAG become available at the next step together with the initial tool set.
If ToolRAG is called in a step, do not execute other graph tools in that same step. Use the newly retrieved tools in the following step.
If an appropriate tool is already available, use it directly instead of calling ToolRAG.

EVIDENCE USE

Base the answer only on evidence retrieved from the experimental protocol graph.
Do not introduce experimental steps, entities, relations, parameters, values, or other details that are not supported by the retrieved evidence.

MEMORY

Memory contains the accumulated reasoning summaries, tool invocations, and observations from previous steps:
{memory}
Use Memory to:
• preserve information obtained in previous steps;
• guide the next reasoning step;
• connect evidence retrieved across different reasoning steps and tool calls;
• avoid repeating completed retrievals; and
• determine what information is still missing.

OUTPUT FORMAT

At each intermediate step, output exactly one JSON object.
For a normal tool-use step:

```
{"thought": "", "actions": [{"tool": "", "args": {...}}]}
```

Multiple independent tool calls may be included in actions when they can be executed in parallel.
For a ToolRAG step:

```
{"thought": "", "actions": [{"tool": "ToolRAG", "args": {"need": [""], "k": 6}}]}
```

When sufficient evidence has been collected:

```
{"thought": "", "stop": {"answer": "", "support": [""]}}
```

Output only the JSON object for the current step.
Do not include additional text outside the JSON object.

CURRENT QUERY

User query:
{query}

MEMORY

{memory}

Figure S8: Full Graph-Aware GARFIELD Agent prompt used at inference time. The prompt defines the iterative reasoning-and-tool-use procedure over the Multi-View Experimental Protocol Graph, including use of the Graph Schema Card, selection and execution of graph tools, dynamic tool

discovery through TOOLRAG, accumulation of retrieved evidence in Memory, and termination with a final graph-grounded answer.

Graph Schema Card

HIERARCHICAL VIEW

Entities

• **Protocol:** The complete experimental procedure and the root entity of the hierarchy.

• **Process:** A major stage of the Protocol formed by a contiguous sequence of Steps that collectively accomplish an intermediate experimental objective.

• **Step:** An executable procedural instruction within the Process. A Step is performed before proceeding to the next Step.

• **Action:** An indivisible laboratory operation within a Step, defined by a single operative verb and its associated context, such as materials, instruments, parameters, location, etc.

Relations

• `(Protocol)-[HAS_PROCESS]->(Process):` Connects a Protocol to the major Processes that compose the experimental procedure.

• `(Process)-[HAS_STEP]->(Step):` Connects a Process to the Steps that form that stage of the procedure.

• `(Step)-[HAS_ACTION]->(Action):` Decomposes a Step into its constituent atomic Actions.

EXECUTION VIEW

Entities

• **Process**

• **Step**

• **Action**

• **Decision:** A branch or choice point in the workflow that determines which Process, Step, or Action is followed under a particular condition or alternative.

Relations

• `(Process)-[NEXT]->(Process|Decision):` Connects a Process to the next Process or Decision in the workflow.

• `(Step)-[NEXT]->(Step|Decision):` Connects a Step to the next Step or Decision in the workflow.

• `(Action)-[NEXT]->(Action|Decision):` Connects an Action to the next Action or Decision in the workflow.

• `(Decision)-[IF_TRUE]->(Process|Step|Action):` Connects a Decision to the Process, Step, or Action executed when its condition evaluates to true.

• `(Decision)-[IF_FALSE]->(Process|Step|Action):` Connects a Decision to the Process, Step, or Action executed when its condition evaluates to false.

• `(Decision)-[ALTERNATIVE]->(Process|Step|Action):` Connects a Decision to one of several mutually exclusive Processes, Steps, or Actions.

• `(Decision)-[OPTIONAL]->(Process|Step|Action):` Connects a Decision to a Process, Step, or Action that is optional rather than required.

• `(Process|Step|Action)-[REPEATS]->(Process|Step|Action):` Indicates that a Process, Step, or Action is repeated at the corresponding level of the workflow.

• `(Process|Step|Action)-[OVERLAPS_WITH]->(Process|Step|Action):` Indicates that two Processes, Steps, or Actions execute concurrently or overlap at the corresponding level of the workflow.

ACTION CONTEXT VIEW

Entities

• **Action**

Experimental Objects (Material, Biological Sample, Instrument, Output)

• **Material:** A non-biological substance used in an experimental procedure, such as a buffer, chemical, solvent, stain, or medium, excluding biological materials and equipment or containers.

• **BiologicalSample:** The biological specimen under study, such as cells, tissue, or another biological preparation.

• **Instrument:** A device, software tool, container, or other piece of equipment used during an Action.

• **Output:** An experimental object produced or transformed by an Action, such as a pellet, supernatant, eluate, lysate, or other product.

• **Parameter:** A quantitative fact associated with an Action or an experimental object, such as duration, temperature, speed, volume, concentration, or mass.

• **Location:** Represents where an Action occurs or where an experimental object is located.

• **Method:** A named experimental technique or assay, such as PCR or Western blot, rather than an ordinary laboratory verb.

• **Modifier:** Describes the manner in which an Action is performed, such as *gently* or *briefly*.

• **Required Condition:** A qualitative condition required for execution of an Action, such as *on ice*, *sterile*, or *RNase-free*.

• **Rationale:** Represents the stated purpose or cause of an Action and captures why the operation is performed.

• **Recipe:** Represents the composition of a mixture, reagent, or buffer.

• **Property:** Represents an intrinsic characteristic of an experimental object, such as *light-sensitive* or *carcinogenic*.

Relations

• `(Action)-[USES]->(Material|Instrument|BiologicalSample):` Connects an Action to the experimental objects that it consumes, handles, or operates on.

• `(Action)-[PRODUCES]->(Output):` Connects an Action to an Output that it creates or transforms.

• `(Action)-[HAS_PARAMETER]->(Parameter):` Associates an Action with a quantitative execution condition, such as time, temperature, or speed.

• `(Material|Instrument|BiologicalSample|Output)-[HAS_PARAMETER]->(Parameter):` Associates an experimental object with a quantitative attribute, such as volume, concentration, or mass.

• `(Action)-[USES_METHOD]->(Method):` Associates an Action with the named experimental technique or assay used to perform it.

• `(Action)-[SITE]->(Location):` Specifies where the Action itself is performed.

• `(Material|Instrument|BiologicalSample|Output)-[LOCATED_AT]->(Location):` Specifies where an experimental object is located or placed.

• `(Action)-[HAS_MODIFIER]->(Modifier):` Associates an Action with information describing how it is performed.

• `(Action)-[HAS_RATIONALE]->(Rationale):` Associates an Action with its stated experimental purpose or cause.

• `(Action)-[HAS_REQUIRED_CONDITION]->(RequiredCondition):` Associates an Action with a qualitative condition required for its execution.

• `(Material|Instrument|BiologicalSample|Output)-[HAS_RECIPE]->(Recipe):` Associates an experimental object with its composition or preparation recipe.

• `(Material|Instrument|BiologicalSample|Output)-[HAS_PROPERTY]->(Property):` Associates an experimental object with one of its properties.

CROSS-PROTOCOL ALIGNMENT VIEW

Entities

• **Action**

- **Material**
- **BiologicalSample**
- **Instrument**
- **Output**
- **Canonical Material | BiologicalSample | Instrument | Output:** Shared normalized representations that group equivalent protocol-specific experimental objects.
- **Canonical Action:** A shared normalized representation that groups Actions corresponding to the same underlying experimental operation.

Relations

`(Material|Instrument|BiologicalSample|Output) -[INSTANCE_OF]->(canonical Material|Instrument|BiologicalSample|Output)`
Connects each protocol-specific experimental object to its corresponding shared canonical representation.

`(Action)-[INSTANCE_OF]->(canonical Action)`
Connects each protocol-specific Action to its corresponding shared canonical Action.

Figure S9: Graph Schema Card provided to the GARFIELD agent for reasoning over the Multi-View Experimental Protocol Graph Representation. The card specifies the entities and relations available across the hierarchical, execution, action-context, and cross-protocol alignment views, enabling the agent to identify where query-relevant information is represented and how it can be traversed or retrieved using graph tools.

Table S14: Graph-tool library available to the graph-aware GARFIELD agent. Each tool exposes a structured operation over the Multi-View Experimental Protocol Graph, including concept and action retrieval, local graph traversal, protocol-level access, cross-protocol comparison, procedural verification, and step or section ordering. The agent dynamically selects among these tools during iterative reasoning and can retrieve additional tools through TOOLRAG when the required capability is not currently exposed.

Tool	Description
`ToolRAG`	Retrieves tools from the available tool library based on a natural-language description of the capability needed. It returns the most relevant tools together with their descriptions, input arguments, and usage details, allowing the agent to dynamically identify and use appropriate tools.
`Resolve_Concept`	Maps a free-text name of a material, instrument, biological sample, or action to its canonical concept in the graph by resolving synonyms, spelling variations, and alternative names.
`Retrieve_Actions`	Retrieves protocol actions that match a structured pattern defined by the action verb, involved experimental objects, and parameter values. It ranks candidate actions by how well they match the provided facets.
`Get_Action`	Retrieves the complete structured information for a specific protocol action using its action ID. It returns all associated action context and is used when a particular action has already been identified and its full details need to be examined or verified.
`Locate_by_Context`	Identifies the specific protocol and target action by matching the surrounding procedural context to steps in the graph. It is used when the action itself is ambiguous, and the context given with the query provides the information needed to determine the correct protocol and action.
`Traverse`	Retrieves nodes or actions directly connected to a known graph node through a specified relationship, such as the next or previous action, used or produced entities, repeated or overlapping actions, or the enclosing step, process, or protocol. It is used to follow procedural relationships and retrieve information that is structurally connected to an already identified action, step, or process.
`Get_Control_Flow`	Retrieves the execution view associated with a protocol, process, or step, including conditional, alternative, and optional paths. It is used to determine how the procedure changes under different conditions or which step should be followed in an if/then scenario.
`Find_Protocols`	Identifies protocols relevant to a free-text experimental objective by searching protocol titles and descriptions.

Tool	Description
Get_Protocol	Retrieves the high-level metadata and structure of a protocol, including its title, abstract, keywords, number of steps, and major processes. It is used to understand the protocol's overall purpose and organization without retrieving detailed experimental values.
Get_Protocol_Resources	Retrieves the complete set of materials, bio samples, and instruments used throughout a specific protocol. It is used to identify all experimental resources required to execute the protocol.
Resource_Consensus	Aggregates materials and instruments across protocols relevant to a given experimental objective and identifies which resources are commonly used, together with how consistently they appear across the retrieved protocols.
Slot_Consensus	Identifies the most supported value for a missing or uncertain part of an experimental action by comparing similar actions across protocols. Given the surrounding context and the type of information being sought, it aggregates candidate values for the target slot, such as the material, instrument, biological sample, output, action, or parameter, and provides their frequency and supporting examples.
Describe_Concept	Retrieves a detailed profile of a canonical concept, including its definition, ontology information, usage across protocols, associated actions, co-used entities, typical parameter ranges, and produced outputs. It is used to understand what a material, instrument, biological sample, or action represents in the graph and how it is typically used across the protocol corpus.
Concept_Neighbors	Identifies canonical concept nodes related to a given canonical concept by retrieving concepts that are commonly used with it or belong to the same ontology parent.
Search_StepText	Performs lexical search over the original text of individual protocol steps and returns steps whose wording best matches a free-text query.
Find_Matching_Steps	Compares a query against existing protocol steps and retrieves the most semantically similar steps, together with their structured actions and parameter values.
Find_Structure_Analogous_Steps	Identifies experimentally similar steps in other protocols based on the provided action, experimental objects, and context.
Action_Transitions	Identifies which canonical actions most commonly occur immediately before or after a specified action across the protocol corpus. It is used to infer which operation is most likely to precede or follow a given action in an experimental workflow.
Verify_Step_Series	Evaluates whether the specified action, experimental objects, or context is consistent with analogous action sequences across other protocols. It uses the target action together with its preceding and following action to compare the stated component against corresponding components observed in similar sequences.
Find_Matching_Step_Order	Identifies the protocol whose steps best match a given set of steps and returns those steps in the order recorded in that protocol.
Find_Matching_Process_Order	Identifies the protocol whose processes best match a given set of process titles and returns those processes in the order recorded in that protocol.
Order_Sections_By_Series	Orders sections by using their typical positions across the protocol corpus together.
Order_By_Action_Flow	Orders sections based on the canonical actions associated with each title and the positions where those actions typically occur across protocols.

Tool	Description
`Order_By_Analogy`	Orders sections by finding similar protocols, reading each similar protocol's actual section order, and voting on pairwise precedence relationships. It is the nearest-analog approach rather than a population-average approach.