BioTool: A Comprehensive Tool-Calling Dataset for Enhancing Biomedical Capabilities of Large Language Models

Xin Gao[1]* **Ruiyi Zhang**[1]* **Meixi Du**[1] **Peijia Qin**[1] **Pengtao Xie**[1,2]†

[1]UC San Diego [2]MBZUAI
{xig022, ruz048, p1xie}@ucsd.edu

Abstract

Despite the success of large language models (LLMs) on general-purpose tasks, their performance in highly specialized domains such as biomedicine remains unsatisfactory. A key limitation is the inability of LLMs to effectively leverage biomedical tools, which clinical experts and biomedical researchers rely on extensively in daily workflows. While recent general-domain tool-calling datasets have substantially improved the capabilities of LLM agents, existing efforts in the biomedical domain largely rely on in-context learning and restrict models to a small set of tools. To address this gap, we introduce BIOTOOL, a comprehensive biomedical tool-calling dataset designed for fine-tuning LLMs. BIOTOOL comprises 34 frequently used tools collected from the NCBI, Ensembl, and UniProt databases, along with 7,040 high-quality, human-verified query–API call pairs spanning variation, genomics, proteomics, evolution, and general biology. Fine-tuning a 4-billion-parameter LLM on BIOTOOL yields substantial improvements in biomedical tool-calling performance, outperforming cutting-edge commercial LLMs such as GPT-5.1. Furthermore, human expert evaluations demonstrate that integrating a BIOTOOL-fine-tuned tool caller significantly improves downstream answer quality compared to the same LLM without tool usage, highlighting the effectiveness of BIOTOOL in enhancing the biomedical capabilities of LLMs. The full dataset and evaluation code are available at https://github.com/gxx27/BioTool.

1 Introduction

The rapid advancement of large language models (LLMs) has revolutionized natural language processing, enabling unprecedented performance across a wide range of general-purpose tasks (OpenAI, 2023; Bai et al., 2023). However, their capabilities in biomedical domains remain limited, which hinders their deployment in high-stakes, real-world biomedical applications (Chen et al., 2025; Li et al., 2025a). A key reason for this limitation is the insufficient ability of LLMs to effectively leverage specialized biomedical tools (Jin et al., 2024). Unlike commonsense questions that can often be answered directly, biomedical problems typically require even expert researchers to consult external tools and databases before drawing reliable conclusions (NCBI, 2017). For instance, even for human biologists, the biological function of a raw nucleotide sequence cannot be reliably inferred without the aid of computational tools, such as BLAST or other sequence similarity–based methods (Altschul et al., 1990). As shown in Figure 1, LLMs that lack access to or integration with such tools are therefore prone to hallucinations and imprecise generalizations, undermining their reliability for scientific discovery.

Given these challenges, early attempts have integrated biomedical and chemistry tools into LLMs via in-context learning (Jin et al., 2024; Bran et al., 2024). Although these approaches show improvements, they are constrained to a small set of available tools due to limited context length. Moreover, biomedical research tools often support diverse and complex usage scenarios that cannot be fully captured by a few lines of textual prompts, which hinders LLMs from fully realizing their potential in biomedical tool usage. Furthermore, they require models to map natural-language questions to highly specialized schemas, identifiers, and parameter conventions to reliably retrieve biologically relevant evidence. Inspired by the success of instruction-tuning–based tool-calling datasets in the general NLP domain (Liu et al., 2024; Patil et al., 2024), we address this gap by curating a comprehensive biomedical tool-calling dataset, BIOTOOL.

BIOTOOL is an instruction fine-tuning–style biomedical tool-calling dataset consisting of 7,040

* Equal contribution.
† Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 46071–46086

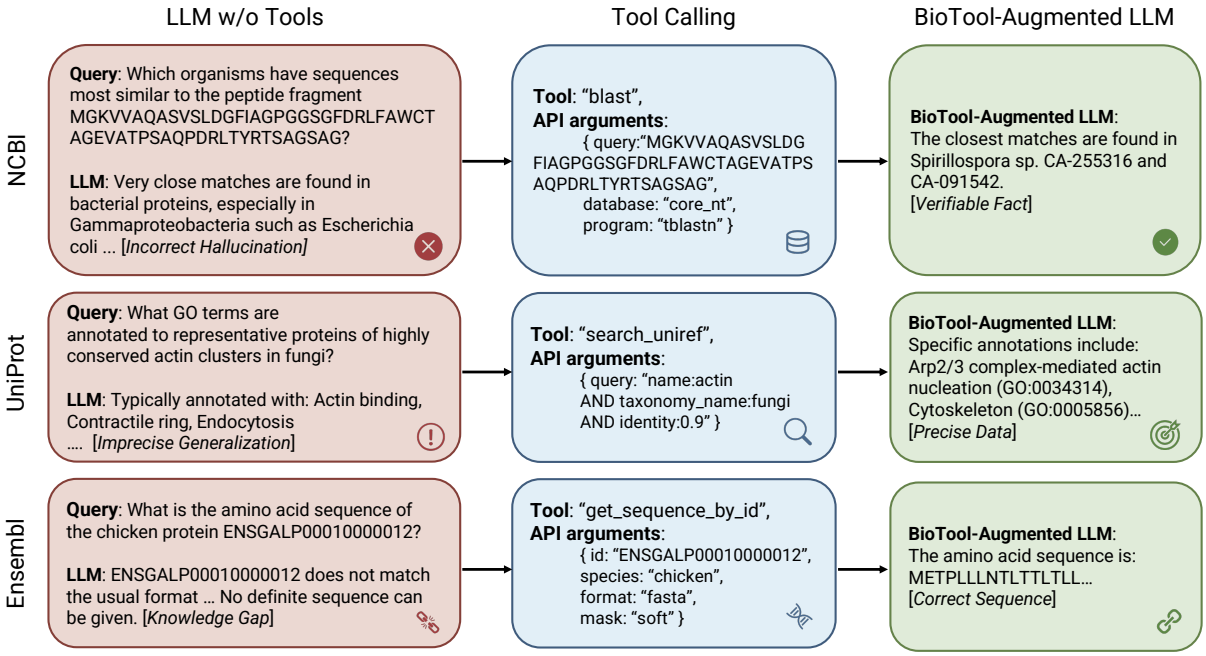

Figure 1: Comparison between answers generated by LLMs without tools and BIOTOOL-augmented LLMs for biomedical queries. LLMs without tools often hallucinate or produce imprecise answers (left), whereas BIOTOOL-augmented LLMs (right) generate API calls and retrieve critical information from biomedical databases, leading to higher-quality responses.

high-quality, human-verified query–API call pairs. It includes 34 frequently used tools from the NCBI (NCBI, 2017), Ensembl (Hubbard et al., 2002), and UniProt (The UniProt Consortium, 2017) databases, spanning multiple subdomains such as variation, genomics, proteomics, evolution, and general biology. To construct the dataset, we first manually select 34 tools from NCBI, Ensembl, and UniProt that are widely used in biomedical research. We then collect official documentation for these tools from their respective websites and use them to generate diverse combinations of API parameters with the assistance of LLMs. The synthesized API calls are executed and filtered to remove cases with unavailable or uninformative responses, resulting in 3,829 unique API calls. Next, we prompt cutting-edge reasoning models (OpenAI, 2025) with these API calls and their corresponding responses to generate potential user queries. These queries are subsequently evaluated by an LLM-based judge to assess whether the API responses meaningfully support answering the queries, followed by a final round of human expert review focusing on biological relevance and correctness. This process yields 7,040 high-quality query–API call pairs, which is the final BIOTOOL dataset.

We evaluate the quality and effectiveness of BIOTOOL through two sets of experiments. First, we fine-tune several open-source LLMs with 4B to 8B parameters on the BIOTOOL training split and compare them with cutting-edge commercial LLMs, including GPT-5.1, Gemini-3 Pro, and Claude-4.5-Sonnet, using in-context learning. Results on the test split show that smaller LLMs fine-tuned with BIOTOOL significantly outperform commercial LLMs with hundreds of times more parameters in terms of tool-calling quality. For example, a BIOTOOL-fine-tuned 4B Qwen-3 model outperforms the best-performing Claude-4.5-Sonnet by 15.0% in overall API-calling quality. Second, we conduct human evaluations to assess whether BIOTOOL-enhanced LLMs produce higher-quality answers from the perspective of biomedical researchers. On 1,048 test queries, a GPT-5.1 model augmented with oracle BIOTOOL API calls achieves 88.4% higher normalized answer quality compared to the same model without tool usage, demonstrating the intrinsic quality of the BIOTOOL dataset. Moreover, a GPT-5.1 model augmented with a BIOTOOL-fine-tuned API caller achieves 69% higher normalized answer quality compared to the raw GPT-5.1 model, highlighting the effectiveness of BIOTOOL in training tool-using LLMs and enhancing their biomedical capabilities.

2 Related Works

Early general-purpose tool-calling models, such as Toolformer (Schick et al., 2023) and Gorilla (Patil et al., 2024), established that LLMs can be trained to invoke external APIs, thereby grounding responses in retrieved data to mitigate hallucinations. Subsequent frameworks like ToolBench (Qin et al., 2023) and APIGen (Liu et al., 2024) advanced this capability by introducing scalable pipelines for generating synthetic instruction-tuning data. Despite these advancements, generalist models often struggle with specialized scientific domains like biomedicine because they rely on broad datasets that include only a negligible fraction of corresponding tools and frequently fail to adhere to the rigorous schema constraints of scientific databases. To address these limitations, domain-specific agents have emerged. GeneGPT (Jin et al., 2024) pioneered this shift by utilizing in-context learning (Wei et al., 2023) to enable access to NCBI Web APIs. Similarly, systems such as SciAgent (Li et al., 2025b) and ChemCrow (Bran et al., 2024) have successfully integrated tool-augmented agents for complex reasoning in scientific and chemical research. While more recent entries like Biomni (Huang et al., 2025) have introduced general-purpose agents for biomedical tasks, they primarily focus on a restricted subset of tools. Consequently, they lack the comprehensive, full-list interface to primary authoritative biomedical databases.

3 The BioTool Dataset

This section details the development and composition of BIOTOOL. We first present an example data entry from BIOTOOL to illustrate the structure of a query–API call pair. Each entry includes a *user query* field, which contains a realistic clinical or biomedical question expressed in free-form text. The *tool information* field provides descriptions of the tools required to answer the query, while the *API arguments* specify the input parameters for the corresponding API endpoint. Executing the API endpoint with these arguments returns an *observations*, which contains information used to augment the LLM's response. We note that the observation is fully determined by the API endpoint and its arguments; it is included in the dataset for completeness and user convenience.

Next, we describe the sequential construction pipeline used to generate and verify biomedical tool calling pairs in Section 3.1, illustrated in Figure 2. We then provide a quantitative analysis of the resulting dataset, highlighting its functional utility and biological diversity in Section 3.2.

Example of BioTool Data Entry

User Query

```
Could you provide concise definitions for the major severe immunodeficiency disorders?
```

Tool Information

```
Database:     "UniProt"
Tool:         "human_diseases"
API Endpoint: "search_human_diseases"
```

API Arguments

```
query:  "name: immunodeficiency AND name: severe"
fields: "definition"
sort:   "id asc"
```

Observations

```
(id: "DI-00171", definition: "An autosomal recessive immunologic disorder characterized by the loss of expression of MHC class II antigens on antigen-presenting cells..."),
(id: "DI-00305", definition: "A form of chronic granulomatous disease..."),
...
```

3.1 Dataset Construction Pipeline

Tool Selection We select three major online API providers: the National Center for Biotechnology Information (NCBI), UniProt, and Ensembl as the tool source for BIOTOOL, motivated by their roles as the authoritative repositories within the global biomedical research infrastructure (Sayers, 2010; Ahmad et al., 2025; Yates et al., 2014). These three platforms are widely considered the definitive standard because they offer expansive and highly interoperable data spanning the entire central dogma of biology, encompassing the full spectrum from raw genomic sequences to functional protein annotations.

Across the three databases, we comprehensively review their websites and manually select tools that are critical for answering biomedical and clinical questions. During this process, we exclude tools with limited biomedical relevance (e.g., APIs that only return service or versioning information) as well as deprecated or unstable tools. As a result, we curate a diverse set of 34 tools comprising 124 API endpoints, each of which is frequently used

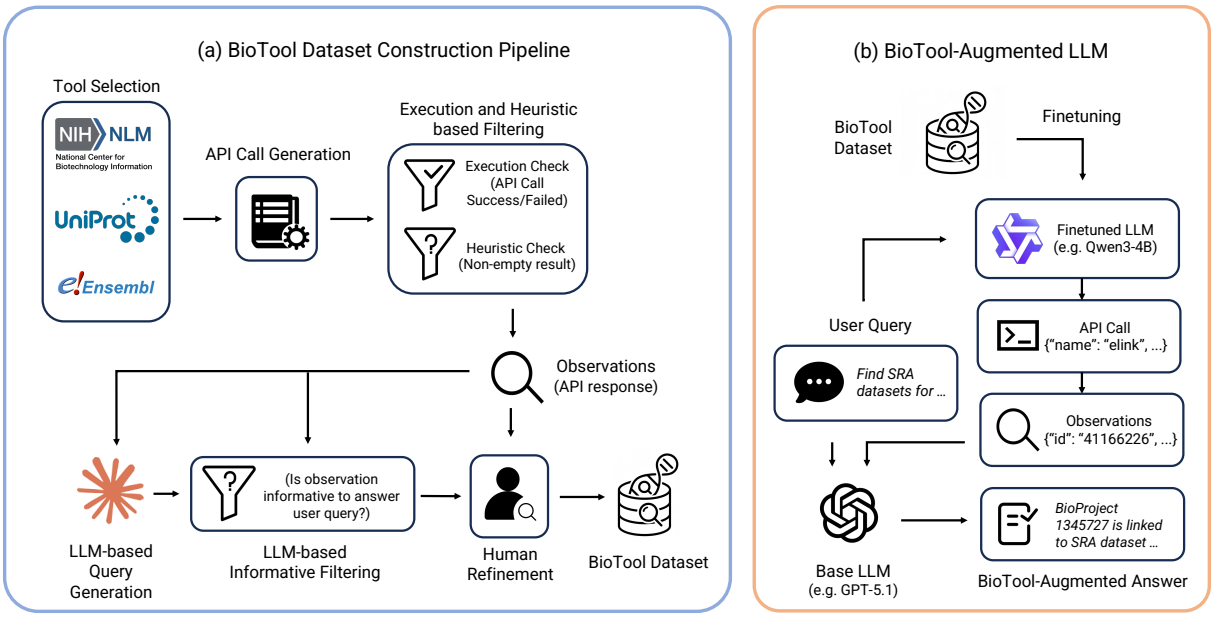

Figure 2: The systematic workflow of BIOTOOL spans from automated dataset construction to downstream application. Panel (a) illustrates the multi-stage construction pipeline, which includes initial tool selection from primary databases, automated API call generation, and a rigorous filtering process involving execution checks, heuristic validation, and LLM-based informativeness assessment. Panel (b) depicts the inference-time application, where specialized API-calling models fine-tuned on BIOTOOL enable base LLMs to retrieve grounded observations and generate verifiable biological answers.

in biomedical research workflows. The complete list of selected tools is provided in Appendix F. In addition, we collect the official documentation for each API endpoint from the corresponding website. These documents specify API usage, input arguments, constraints, and example calls, and serve as essential resources for subsequent stages of API call synthesis and user query generation.

API Call Synthesis and Verification Based on the curated tool set and associated documentation, we manually select critical API arguments corresponding to biologically meaningful identifiers for each API endpoint. These arguments, such as taxon IDs, gene symbols, and UniProt accession numbers, ensure that the synthesized API calls are biologically diverse and scientifically plausible. Given the selected arguments, we follow prior work (Liu et al., 2024) to randomly sample a large set of candidate API calls. These candidates are then executed to filter out cases that result in client errors, timeouts, or empty responses. To further improve data quality, we design a novel heuristic-based filtering strategy to remove API calls that are overly similar to existing ones, as well as those whose returned observations lack biological significance. Details of this heuristic filter are provided in Appendix A. After this verification process, we obtain a collection of 6,391 unique API calls.

User Query Generation Given the synthesized API calls, we leverage cutting-edge LLMs to generate corresponding user queries, following a self-instruct–style paradigm established in prior work (Wang et al., 2022; Patil et al., 2024; Liu et al., 2024). Specifically, LLMs are prompted with an API call, its documentation, and its corresponding observation, together with a small set of human-crafted in-context query–API call pairs, to generate realistic user queries.

To further improve the quality and biological relevance of BIOTOOL, we introduce two novel adaptations to ensure both the *necessity* and *sufficiency* of the API observations. First, to enforce *necessity*, we apply Chain-of-Thought (CoT) prompting (Wei et al., 2023) using a strong reasoning model (OpenAI o3 (OpenAI, 2025)) when generating user queries. The model is first prompted to summarize the technical details of the API observation into a natural-language description, which is then used to generate the final user query. This procedure ensures that the observation is required to answer the query, while keeping the query realistic and avoiding explicit references to specific tools or API calls. The detailed system and user prompts for this process are provided in Appendix E.1. Second,

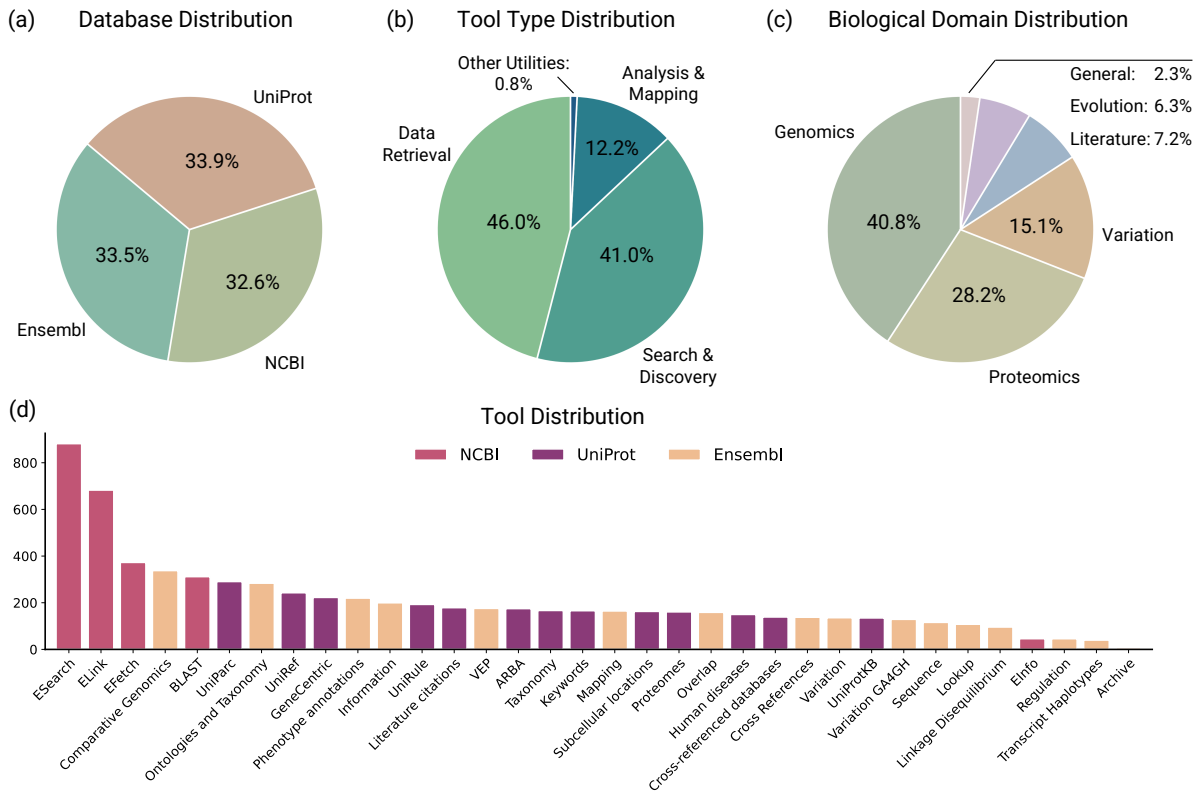

Figure 3: Distribution analysis of the 7,040 samples within BIOTOOL across four dimensions. Panel (a) shows the distribution across source databases. Panel (b) illustrates the distribution of samples by tool type. Panel (c) presents the distribution across various biological domains. Panel (d) delineates the distribution of user queries across the 34 distinct biological tools.

to ensure *sufficiency*, we employ another cutting-edge LLM (Claude Haiku 4.5 (Anthropic, 2025)) to perform informativeness-based filtering, inspired by the LLM-as-a-judge framework (Zheng et al., 2023). The model is prompted to follow a structured rubric and classify a query–API call pair as informative if the observation contains at least one relevant fact or a partial summary that supports the user's intent. Pairs in which the observation is unrelated to the query or too vague to support a concrete response are discarded. The specific judge prompts are provided in Appendix E.2.

Human Refinement The final stage involves a comprehensive manual review conducted by human evaluators with at least a college-level background in bioinformatics. The evaluators first identify and remove low-quality queries. For the remaining samples, they refine pedantic or unnatural phrasing and ensure the accuracy of biological terminology and nomenclature. After this round of filtering and correction, the final BIOTOOL dataset comprises 7,040 high-quality samples.

This instruction fine-tuning–style dataset is primarily used to train open-source LLMs as API-calling models, following training paradigms established in general-domain tool-calling datasets (Patil et al., 2024; Liu et al., 2024). A BIOTOOL-trained LLM can assist state-of-the-art LLMs in generating grounded and scientifically accurate responses, as illustrated in the right panel of Figure 2.

3.2 Data Statistics

The BIOTOOL dataset is derived from 34 distinct biological tools and 124 unique API endpoints, encompassing a wide array of scientific content categorized across several key dimensions. As shown in Figure 3(a), the distribution of tools across databases is well balanced, with comparable proportions from NCBI, UniProt, and Ensembl. Figure 3(b) illustrates the diversity of tool types included in BIOTOOL, ranging from data retrieval (e.g., nucleotide identifiers fetching) and search and discovery (e.g., phenotype-based gene discovery) to biological analysis and mapping (e.g., cross-referencing SNP identifiers). Figure 3(c) highlights the dataset's broad scientific scope, covering domains such as genomics (e.g.,

gene tree querying), proteomics (e.g., protein sequence alignment), variation analysis (e.g., linkage disequilibrium analysis), and evolutionary biology (e.g., species-level taxonomy identification). Finally, Figure 3(d) shows that BIOTOOL includes both frequently accessed general-purpose tools and a long tail of specialized tools, all of which are essential for complex scientific discovery across the central dogma.

4 Experimental Results

To evaluate the effectiveness of BIOTOOL, we first compare the API-calling capabilities of small open-source LLMs fine-tuned on BIOTOOL against their vanilla counterparts and cutting-edge proprietary LLMs using in-context learning. We then conduct human expert evaluations to compare the answer quality of baseline LLMs with that of BIOTOOL-augmented LLMs.

4.1 Experimental Setup

BioTool score We define a BioTool performance score to automatically evaluate the capability of an LLM as an API caller on the BIOTOOL dataset, especially the alignment of retrieved information with the user's intent. Specifically, assume we have the test set $D = \{(q_1, o_1), ...(q_n, o_n)\}$, where q_i is the i^{th} user query and o_i is the observation obtained from ground-truth API calling in the dataset. The BioTool score on this test set $\mathrm{S}(D)$ for a LLM API caller f is then defined as follows:

$$\mathrm{S}(D) = \sum_{i=1}^{n} \mathrm{Sim}\big(f(q_i), o_i\big) \tag{1}$$

where $\mathrm{Sim}(\hat{o}, o)$ computes the semantic embedding similarity of two text strings: the ground truth observation o and the corresponding observation $\hat{o}$ from LLM API caller prediction. In practice, we use a MedCPT model (Jin et al., 2023) to get a sentence embedding for an observation. API calls may fail due to incorrect model generation, yielding an empty string $\hat{o} = \varepsilon$. In this case, we set $\mathrm{Sim}(\varepsilon, o) = 0$. Intuitively, this score determines model performance by measuring whether the retrieved biological facts remain semantically similar to the required information, even when the technical implementation of the call differs from the reference.

Additional Metrics Based on the BioTool score, we define two additional metrics to further characterize model performance. Similar metrics have been widely adopted in existing API-calling benchmarks (Patil et al., 2025). Firstly, we define API calling success rate AS as follows:

$$\mathrm{AS}(D) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big[\mathrm{Sim}\big(f(q_i), o_i\big) > 0\big] \tag{2}$$

where $\mathbf{1}[\cdot]$ is the indicator function. A zero similarity indicates API calling failure due to incorrect formatting, invalid API names, or improper parameter values. Conceptually, this metric focuses on the model's capability to generate API calls that execute correctly and return a valid response containing data. Secondly, we define a exact match score EM as follows:

$$\mathrm{EM}(D) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\big[\mathrm{Sim}\big(f(q_i), o_i\big) = 1\big] \tag{3}$$

which measures the proportion of predictions whose resulting observations exactly match the ground-truth reference observation, requiring the model to correctly identify the API endpoint and provide all required parameters with values that exactly match the reference.

Models In this study, we use four cutting-edge proprietary models, including GPT-5.1, GPT-5.1-Codex, Gemini 3 Pro, and Claude 4.5 Sonnet (OpenAI, 2025b,a; Google, 2025; Anthropic, 2025) under an in-context learning scheme. We use four open-source models, which are Llama3.1-8B-Instruct, Qwen3-8B, Qwen2.5-7B-Instruct, and Qwen3-4B-Instruct (Grattafiori et al., 2024; Yang et al., 2025; Qwen et al., 2025), for both in-context learning and BioTool-based fine-tuning. We report the average performance across three independent runs.

4.2 Results on Tool Calling Capability

In this section, we first fine-tune small open-source models on the training split of the BIOTOOL dataset, which is randomly split under a four-to-one ratio. We use the cutting-edge proprietary model and base open-source models as baselines, and the evaluation for all models was conducted equally on the held-out test set consisting of 1,408 samples in terms of BioTool score. As shown in Table 1, there is a clear performance advantage for BIOTOOL-fine-tuned models over much larger LLMs under in-context learning. The fine-tuned

Model	NCBI	UniProt	Ensembl	Overall
Proprietary Models				
GPT-5.1	15.5	78.7	72.8	55.4
GPT-5.1-Codex	14.8	80.5	74.5	56.3
Gemini 3 Pro	72.5	87.8	77.3	79.2
Claude 4.5 Sonnet	79.8	87.3	77.0	81.4
Base Open-Source Models				
Llama3.1-8B-Ins	28.4	77.5	58.8	54.8
Qwen3-8B	26.2	73.1	63.5	54.1
Qwen2.5-7B-Ins	48.0	79.0	65.3	64.0
Qwen3-4B-Ins	53.4	71.9	64.0	63.1
BioTool-fine-tuned Models				
Llama3.1-8B-Ins	91.1	88.9	93.7	91.2
Qwen3-8B	89.5	84.8	83.9	86.1
Qwen2.5-7B-Ins	91.8	87.6	88.0	89.2
Qwen3-4B-Ins	**93.7**	**91.0**	**96.3**	**93.6**

Table 1: Comparative evaluation of models on the BIOTOOL dataset, measured by the BioTool score (higher is better). Scores are reported for each constituent database (NCBI, UniProt, Ensembl) and overall. Model names with the suffix *Ins* denote instruction-tuned variants. Bold values indicate the best performance in each column.

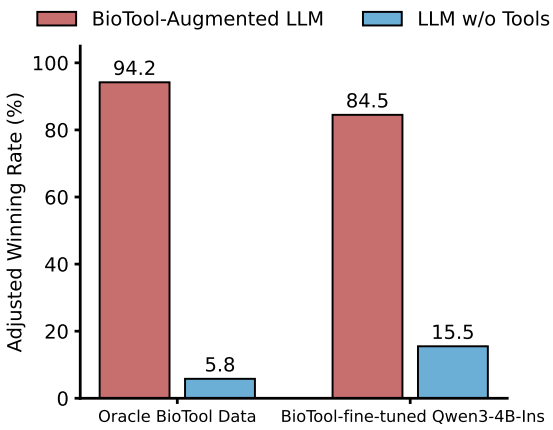

Figure 4: Human evaluation results comparing answer quality between BIOTOOL-augmented LLMs and LLMs without tool usage, using GPT-5.1 as the base model. The augmented settings include GPT-5.1 with oracle BIOTOOL data (left) and GPT-5.1 with a BIOTOOL-fine-tuned Qwen3-4B-Instruct tool caller (right).

4B model achieved the highest overall BioTool score, representing a 15.0% improvement over the strongest proprietary model, Claude 4.5 Sonnet, and 68.9% higher performance than GPT-5.1. This gap suggests that the general-purpose pre-training of frontier LLMs together with in-context learning is insufficient to navigate the specialized technical constraints and precise parameter mappings of biological repositories. Instead, the high-density training signals within the BIOTOOL dataset allow significantly smaller models to acquire the necessary domain expertise that remains elusive to even the largest proprietary models.

4.3 Human Evaluation of Answer Quality

The ultimate criterion for assessing the usefulness of a tool-calling dataset is its ability to improve the quality of LLM-generated answers. To evaluate this, we use GPT-5.1 as the base model and compare its performance under three settings: (1) no tool augmentation, (2) augmentation with ground-truth BIOTOOL API calls, and (3) augmentation with a BioTool-fine-tuned Qwen3-4B-Instruct tool-calling model. We evaluate these three settings across all test queries using side-by-side human judgments by two annotators with college-level bioinformatics backgrounds. Annotators compare settings (1) vs. (2) and (1) vs. (3), selecting the better answer based on informativeness and task fulfillment, while rejecting vague or scientifically incorrect responses. The normalized win rates for the two comparisons are shown in Figure 4. The reported win rates are the average of the two annotators' individual results. Raw preference results and normalization procedures are detailed in Appendix D.

We observe that tool augmentation substantially improves the quality of biomedical answers, demonstrating that grounding LLMs in verifiable data from NCBI, Ensembl, and UniProt effectively mitigates domain-specific hallucinations and imprecise generalizations. The oracle configuration achieves a 94.2% win rate over the base model, highlighting the high quality of the BIOTOOL dataset. Similarly, the BIOTOOL-fine-tuned Qwen3-4B-Instruct model attains an 84.5% win rate, indicating that a small, fine-tuned model can improve the correctness and helpfulness of large commercial LLMs as judged by human evaluators, further demonstrating the practical utility of BIOTOOL.

4.4 Additional Results

We report results for the additional metrics under the same experimental settings in Table 1 to provide further insights into model behavior and dataset

Model	NCBI		UniProt		Ensembl		Overall	
	EM	AS	EM	AS	EM	AS	EM	AS
Proprietary Models								
GPT-5.1	0.0	16.6	1.7	97.9	8.9	79.4	3.5	64.4
GPT-5.1-Codex	0.0	16.2	1.7	98.3	6.3	80.5	2.6	64.7
Gemini 3 Pro	3.2	76.7	2.1	99.8	16.3	82.0	7.1	86.2
Claude 4.5 Sonnet	3.6	85.5	1.1	**100.0**	15.0	82.4	6.5	89.4
Base Open-Source Models								
Llama3.1-8B-Ins	0.0	33.2	1.0	96.7	5.5	68.9	2.1	66.1
Qwen3-8B	0.5	28.9	1.2	87.8	10.0	70.1	3.8	62.1
Qwen2.5-7B-Ins	0.0	54.6	1.6	97.6	6.3	74.1	2.6	75.4
Qwen3-4B-Ins	1.3	58.5	1.7	88.8	7.8	72.8	3.6	73.3
BioTool-fine-tuned Models								
Llama3.1-8B-Ins	43.1	93.3	7.4	99.9	43.1	96.6	31.2	96.6
Qwen3-8B	40.8	92.3	4.5	99.8	35.6	88.8	26.9	93.7
Qwen2.5-7B-Ins	44.6	93.8	5.4	99.9	42.4	91.5	30.8	95.1
Qwen3-4B-Ins	**66.7**	**94.8**	**9.3**	**100.0**	**51.1**	**98.8**	**42.4**	**97.8**

Table 2: Comparative evaluation of EM and AS metrics (higher is better). Model names with the suffix *Ins* denote instruction-tuned variants. Bold values indicate the best performance in each column.

characteristics. As shown in Table 2, there is a clear divergence between Exact Match (EM) and API Success (AS), particularly for proprietary models. Although models such as Claude 4.5 Sonnet and Gemini 3 Pro achieve high AS scores, their EM remains extremely low, indicating difficulty in producing parameterizations that exactly match reference specifications. In contrast, the BIOTOOL-fine-tuned Qwen3-4B-Instruct achieves an EM nearly six times that of the best proprietary model, highlighting the necessity of fine-tuning for learning the precise syntax of biological APIs. The EM–AS gap also reflects the varying complexity of biological repositories. On the NCBI subset, proprietary models such as GPT-5.1 fail to achieve any exact matches and frequently encounter execution errors, likely due to strict identifier formats and nested parameters. Fine-tuned models, however, maintain high execution success, demonstrating that BIOTOOL trains functionally robust models that produce valid and biologically meaningful API calls even without exact string matches.

4.5 Error Analysis

To gain deeper insight into model failure modes, we conduct a systematic error analysis over all API call failures on the whole test set. We categorize parameter-level mistakes into three mutually non-exclusive types. *Missing parameters* refers to cases where the predicted call omits one or more arguments present in the ground-truth reference, thereby altering the biological scope of the retrieved result; for instance, the omission of the `species` argument from an Ensembl comparative genomics call, which causes the endpoint to return homology data for an unintended reference organism. *Extra parameters* refers to cases where the predicted call includes arguments absent from the reference, potentially redirecting the query's intent; like injecting a `canonical` flag into a VEP annotation call, which restricts consequence reporting to canonical transcripts only and suppresses annotations for non-canonical isoforms that may be biologically or clinically relevant. *Wrong parameter values* refers to cases where the argument name is correct, but the assigned value is semantically incorrect; for example, specifying `"blastp"` in place of `"tblastn"` as the BLAST program, which conflates protein-against-protein and protein-against-translated-nucleotide search modes and yields entirely incompatible results. The distribution of these error types across all evaluated models is reported in Table 3.

As shown in the results, proprietary models and non-fine-tuned open-source models exhibit pervasive failures in semantic parameter mapping, in-

Model	Missing	Extra	Wrong
Proprietary Models			
GPT-5.1	5.0	34.9	28.9
GPT-5.1-Codex	5.2	34.6	30.0
Gemini 3 Pro	5.8	8.2	10.9
Claude 4.5	5.8	4.0	8.2
Base Open-Source Models			
Llama3.1-8B-Ins	14.4	27.6	24.6
Qwen3-8B	12.2	17.0	14.3
Qwen2.5-7B-Ins	12.4	17.0	15.3
Qwen3-4B-Ins	6.1	6.4	8.2
BioTool-fine-tuned Models			
Llama3.1-8B-Ins	1.1	1.1	1.8
Qwen3-8B	0.9	1.1	2.0
Qwen2.5-7B-Ins	1.8	2.1	2.6
Qwen3-4B-Ins	**0.6**	**0.6**	**1.6**

Table 3: Distribution of parameter-level error categories among API failures. **Missing**, **Extra**, and **Wrong** denote the proportion of the total test set attributable to failures involving missing, extra, and wrong-value parameters, respectively. Bold values indicate the lowest error rate in each column.

cluding incorrect database or program selections in NCBI Entrez and BLAST calls, misspecified biological identifiers such as taxonomic names, and erroneous traversal targets in Entrez link operations, all of which cause the retrieved data to be biologically misaligned with the user's intent, regardless of whether the call itself executes successfully. BIOTOOL fine-tuning substantially mitigates these semantic mapping failures, yielding dramatically reduced API failure rates across all trained model variants and confirming that correct biomedical API invocation requires domain-specific schema grounding that neither general-purpose pretraining nor in-context learning can reliably provide.

4.6 Case Study

To illustrate how BIOTOOL enhances LLMs' biomedical capabilities, we present an example comparing the API-calling behavior and tool-augmented answers of a BIOTOOL-fine-tuned Qwen3-4B-Instruct model and Claude 4.5 Sonnet on a complex genomic linkage query. The ICL–based Claude model exhibits low precision in handling domain-specific tool usage rules, directly passing literal natural-language terms such as "goat" and "NextGen" from the user query into the API arguments. In contrast, the BIOTOOL-fine-tuned model correctly maps these concepts to the required API parameters. This example demonstrates that BIOTOOL provides high-quality supervision for learning the implicit rules of biomedical tool usage, enabling reliable natural-language-to-tool mapping that state-of-the-art ICL-based LLMs often fail to achieve.

Comparison of API Call and Response to an Example User Query

User Query

```
Within 29:707234-757234, which variant pairs show strong linkage in the NextGen goat population?
```

BioTool-fine-tuned Qwen3-4B API Call

```
get_ld_region(species="capra_hircus",
region="29:707234-757234",
population_name="NextGen:All",
d_prime=0.8)
```

BioTool-fine-tuned Qwen3-4B Response

```
[ {"variation1": "rs661133063",
"variation2": "rs668584442", "d_prime":
1.0 }, ... ]
```

Claude API Call

```
get_ld_region(species="goat",
region="29:707234-757234",
population_name="NextGen")
```

Claude Response

```
{ "error": "Can not find internal name for
species 'goat' " }
```

Oracle API Call

```
get_ld_region(species="capra_hircus",
region="29:707234-757234",
population_name="NextGen:All", r2=0.5)
```

5 Conclusion

In this work, we introduce BIOTOOL, a comprehensive biomedical tool-calling dataset comprising 7,040 human-verified query-API call pairs spanning 124 biomedical tools. Fine-tuning a 4-billion-parameter LLM on BIOTOOL leads to substantial improvements in API-calling performance, surpassing cutting-edge proprietary LLMs. Furthermore, human evaluations confirm that BIOTOOL-augmented LLMs generate more helpful, informative, and scientifically accurate answers compared to the same base models without tool usage, shedding light on the development of reliable biomedical agents in the future.

Limitations

Despite the performance gains observed with BIOTOOL, several limitations remain. Our current framework focuses exclusively on one-hop tool calling responses. This ignores more complex biological problems that cannot be solved with a single API interaction and instead require multi-hop search results or iterative reasoning across multiple tools. Furthermore, we did not fine-tune an independent, specialized biomedical agent. This architectural choice was necessitated by the extreme context length of raw biological observations, which frequently exceed our resource limitations even after post-processing and summarization. Future work should explore long-context architectures and multi-step reasoning trajectories to better support the most intricate clinical and research workflows.

Acknowledgements

We acknowledge funding support from the National Science Foundation (NSF) under grants IIS-2405974 and IIS-2339216, and from the National Institutes of Health (NIH) under grant R35GM157217.

References

Shadab Ahmad, Leonardo Jose da Costa Gonzales, Emily H Bowler-Barnett, Daniel L Rice, Minjoon Kim, Supun Wijerathne, Aurélien Luciani, Swaathi Kandasaamy, Jie Luo, Xavier Watkins, Edd Turner, Maria J Martin, and the UniProt Consortium. 2025. The uniprot website api: facilitating programmatic access to protein knowledge. *Nucleic Acids Research*, 53(W1):W547–W553.

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. 1990. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410.

Anthropic. 2025. System card: Claude haiku 4.5. System Card.

Anthropic. 2025. System card: Claude sonnet 4.5.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenhang Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, K. Lu, and 31 others. 2023. Qwen technical report. *ArXiv*, abs/2309.16609.

Andres M. Bran, Sean Cox, Oliver Schilter, and 1 others. 2024. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535.

Qiang Chen, Yifan Hu, Xiaohan Peng, and 1 others. 2025. Benchmarking large language models for biomedical natural language processing applications and recommendations. *Nature Communications*, 16:3280.

Google. 2025. Gemini 3 pro model card.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Kexin Huang, Serena Zhang, Hanchen Wang, Yuanhao Qu, Yingzhou Lu, Yusuf Roohani, Ryan Li, Lin Qiu, Junze Zhang, Yin Di, and 1 others. 2025. Biomni: A general-purpose biomedical ai agent. *bioRxiv*, pages 2025–05.

Tim Hubbard, David Barker, Ewan Birney, Graham Cameron, Yong Chen, Lucy Clark, Tony Cox, James Cuff, Val Curwen, Thomas Down, Richard Durbin, Eduardo Eyras, James Gilbert, Matthew Hammond, Lukasz Huminiecki, Arek Kasprzyk, Heikki Lehvaslaiho, Peter Lijnzaad, Chris Melsopp, and 16 others. 2002. The ensembl genome database project. *Nucleic Acids Research*, 30(1):38–41.

Qiao Jin, Won Kim, Qingyu Chen, Donald C Comeau, Lana Yeganova, W John Wilbur, and Zhiyong Lu. 2023. Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 39(11):btad651.

Qiao Jin, Yifan Yang, Qingyu Chen, and Zhiyong Lu. 2024. Genegpt: Augmenting large language models with domain tools for improved access to biomedical information. *Bioinformatics*, 40(2):btae075.

Klaus Krippendorff. 2011. Computing krippendorff's alpha-reliability.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Mingchen Li, Zaifu Zhan, Han Yang, Yongkang Xiao, Huixue Zhou, Jiatan Huang, and Rui Zhang. 2025a. Benchmarking retrieval-augmented large language models in biomedical nlp: Application, robustness, and self-awareness. *Science Advances*, 11(47):eadr1443.

Xuchen Li, Ruitao Wu, Xuanbo Liu, Xukai Wang, Jinbo Hu, Zhixin Bai, Bohan Zeng, Hao Liang, Leheng Chen, Mingrui Chen, Haitian Zhong, Xuanlin Yang, Xu-Yao Zhang, Liu Liu, Jia Li, Kaiqi Huang, Jiahao Xu, Haitao Mi, Wentao Zhang, and Bin Dong. 2025b. Sciagent: A unified multi-agent system for generalistic scientific reasoning. *Preprint*, arXiv:2511.08151.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, and 1 others. 2024. Apigen: Automated pipeline for generating verifiable and diverse function-calling datasets. *arXiv preprint arXiv:2406.18518*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

NCBI. 2017. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 46(D1):D8–D13.

OpenAI. 2023. Gpt-4 technical report.

OpenAI. 2025a. Gpt-5.1-codex-max system card.

OpenAI. 2025b. Gpt-5.1 instant and gpt-5.1 thinking system card addendum.

OpenAI. 2025. Openai o3 and o4-mini system card. System Card.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive apis.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Eric Sayers. 2010. A general introduction to the e-utilities. *Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US)*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

The UniProt Consortium. 2017. Uniprot: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. In *Annual Meeting of the Association for Computational Linguistics*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Andrew Yates, Kathryn Beal, Stephen Keenan, William McLaren, Miguel Pignatelli, Graham R. S. Ritchie, Magali Ruffier, Kieron Taylor, Alessandro Vullo, and Paul Flicek. 2014. The ensembl rest api: Ensembl data for any language. *Bioinformatics*, 31(1):143–145.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A Heuristic Filter Detail

In this section, we provide a more granular explanation of the heuristic filtering strategies employed during the API call synthesis and verification phase.

The specific filtering logic varies across the three integrated databases to account for differences in their API architectures and the nature of the biological data they provide. For UniProt, which primarily provides functional protein annotations and sequence data, we implement a strict deduplication process by filtering out all API calls targeting the same unique identifier, such as a UniRef entry ID or keyword entry ID, within the same tool to prevent the over-representation of specific proteins. Furthermore, we validate every execution result by discarding any responses that return empty lists or "null" search results, thereby ensuring that every retained API call contains at least one valid, non-empty biological observation. Ensembl requires a more nuanced dual-path approach to balance diversity and validity when handling complex genomic coordinates. For endpoints with

a restricted set of valid parameter combinations (defined as fewer than 20), where strict ID deduplication would yield insufficient data, we selectively retain entries where the optional parameters, such as species or variants, are not identical, while query IDs are the same. Conversely, for "rich" APIs with an expansive range of possible inputs, we apply a strategy similar to UniProt by filtering out any samples where the combination of required parameters is identical to an existing entry to prevent the model from over-fitting to specific genomic regions. For NCBI, the strategy is optimized for high-throughput tools and general metadata retrieval. We apply specialized heuristics to the BLAST tool, only retaining parameter combinations that involve unique query sequences and return at least one significant alignment hit, while removing matchless queries that cannot support downstream scientific reasoning. Other NCBI APIs are filtered using a standard heuristic method that removes identical identifier calls and verifies that the retrieved observations remain biologically informative.

B Dataset Scale Analysis

To examine how performance scales with training data volume, we train Qwen3-4B-Instruct (Yang et al., 2025) on six subsets of the BIOTOOL training split, ranging from 10% to 100%. Table 4 presents the improvement over the untuned Qwen3-4B-Instruct baseline, whose overall Exact Match, API Success, and BioTool Score are 3.6, 73.3, and 63.1, respectively. The results show that even the smallest training subset yields substantial gains across all three metrics, confirming that BIOTOOL provides strong supervision from the early stages of scaling. As the training set expands, the gain in BioTool Score increases steadily, while Exact Match continues to improve throughout the full range, indicating that parameter-level precision remains the principal source of additional benefit at larger scales.

C Generalization Capability

To examine whether BIOTOOL-fine-tuned models can generalize beyond the APIs observed during training, we construct a stricter evaluation split based on API identity, such that all samples associated with the same API function are assigned to a single partition, and every API in the test set is unseen during training. We then compare the resulting performance of Qwen3-4B-Instruct against GPT-5.1 and GPT-5.1-Codex. As shown in Table 5, Qwen3-4B-Instruct retains a clear advantage in Exact Match and also achieves the strongest BioTool Score, indicating that BIOTOOL fine-tuning continues to improve the structural fidelity and overall quality of API calling even when evaluation is conducted on previously unseen functions. Compared with the previously reported results under the standard random split, the performance gap becomes smaller in this setting, indicating that generalization to unseen APIs is substantially more challenging than generalization to new instances of previously observed APIs. Nevertheless, the continued advantage of Qwen3-4B-Instruct shows that BIOTOOL fine-tuning yields transferable gains that extend beyond memorization of the training API set.

Training Size	Δ EM	Δ AS	Δ BioTool
10%	18.6	20.7	23.2
20%	27.3	21.3	25.5
40%	33.6	23.0	27.8
60%	36.2	23.1	28.5
80%	37.6	22.9	28.9
100%	**38.7**	**24.2**	**30.4**

Table 4: Overall performance gains of Qwen3-4B-Instruction fine-tuned on different fractions of the BIOTOOL training set, measured as absolute improvements over the base Qwen3-4B-Instruction baseline. Δ EM, Δ AS, and Δ BioTool denote gains in Exact Match, API Success, and BioTool Score, respectively. Bold values indicate the largest gain in each column.

Model	EM	AS	BioTool Score
GPT-5.1	4.2	88.6	76.5
GPT-5.1-Codex	3.8	**94.7**	83.5
Qwen3-4B-Ins	**25.7**	93.0	**84.1**

Table 5: Model performance on the unseen-API evaluation split, where all instances associated with the same API function are assigned to a single partition. EM, AS, and BioTool Score denote Exact Match, API Success, and BioTool Score, respectively. Bold values indicate the best result in each column.

D Human Evaluation Details

This section details the manual side-by-side assessment process and provides the raw preference data used to derive the winning rates reported in Sec-

tion 4.3. A total of 1,408 samples were evaluated for each comparison setting by researchers with biological backgrounds to compare the performance of tool-augmented models against the base GPT-5.1 generator. Table 6 summarizes the distribution of these outcomes, including cases where both models performed well, or both failed to provide a satisfactory answer.

Outcome	Qwen3-4B	Oracle
Total Samples	1,408	1,408
Model A Wins	75.3%	90.3%
Model B Wins	6.3%	1.8%
Both Bad	11.2%	3.3%
Both Good	7.2%	4.5%

Table 6: Raw human preference distribution for pairwise model evaluations. Model A refers to the tool-augmented configuration (Qwen3-4B or Oracle), and Model B refers to the base GPT-5.1 model without tool access.

To provide a balanced comparison that accounts for samples where neither model showed a distinct advantage, we calculated the adjusted winning rate reported in Figure 4 based on the logic of McNemar's test (McNemar, 1947). In this framework, "Both Good" and "Both Bad" responses are collectively treated as ties ($n_c = n_{good} + n_{bad}$) and adjusted by splitting them evenly between the two conditions. Specifically, given the raw preference counts n_a and n_b, the adjusted preference numbers n'_a and n'_b were calculated as $n'_a = n_a + \frac{1}{2}n_c$ and $n'_b = n_b + \frac{1}{2}n_c$.

Annotation Reliability. To validate the quality of the labeling procedure, we computed inter-annotator agreement between the two human annotators on the manually labeled samples per task. Table 7 reports Cohen's κ and Krippendorff's α for each task and for the combined dataset. Both metrics exceed the commonly accepted threshold of 0.667 for "acceptable" agreement (Krippendorff, 2011), and fall within the "substantial" range ($\kappa \geq 0.61$) under the Landis & Koch scale (Landis and Koch, 1977), indicating reliable annotation across all settings.

Model	N	%Agree	κ	α
Oracle	1,408	88.6%	0.800	0.800
Qwen3-4B	1,408	82.0%	0.722	0.722

Table 7: Inter-annotator agreement between two human annotators. Each comparison is between a tool-augmented model (Oracle or Qwen3-4B) and the base GPT-5.1 model without tool access. Agreement is computed on the four raw preference categories.

E Prompts

E.1 Prompt for creating user queries

The following prompt is used to generate natural language user queries. It requires four distinct input streams: (1) the source document context, (2) API function specifications, (3) retrieved biological observations, and (4) in-context few-shot demonstrations.

System Prompt

You generate realistic biomedical questions that researchers naturally ask.

TASK OVERVIEW: TWO-PHASE REASONING

Phase 1 – ANALYZE: Map technical parameters to natural language concepts (Qualitative Mapping).
Phase 2 – GENERATE: Create TWO questions (one Broad/Implicit, one Specific/Qualitative).

PHASE 1: PARAMETER MAPPING & ABSTRACTION

Do not simply list parameters. You must translate *Data* into *Language*.

- **VERBATIM (Keep Exact):** Unique identifiers (gene symbols, rsIDs, accessions), Raw sequences (FASTA format), and Coordinates (e.g., "chr1:100-200").
- **QUALITATIVE MAPPING (Translate Numbers/-Codes):**
  - **Thresholds:** Map high numbers to adjectives like "strong" or "significant" (e.g., `d_prime=0.8` → "strong linkage").
  - **Complex Codes:** Simplify technical strings to common terms (e.g., `1000GENOMES...` → "1000 Genomes data").
- **IMPLICIT DEFAULTS (Selectively Omit):** If a parameter just ensures usable results (e.g., `format=json`), OMIT it. The user implies "good results" by asking.

PHASE 2: QUESTION GENERATION STRATEGY

Your goal is **Tool Bias**: The question should be specific enough that *this tool* is the logical choice, without naming it.

Question 1: The "Implicit" Question (Natural & Broad). A question a biologist asks a colleague. Hide strict parameters; assume the tool's filters represent the broad intent.

Question 2: The "Qualitative" Question (Specific Demand). A researcher asking for a specific *quality*. Use adjectives to reflect parameter values (e.g., "strong LD").

RULES:

- Ask only **one** question at a time.
- Keep the question concise and to the point.
- Do not use parentheses for supportive information.

OUTPUT FORMAT
Return JSON only with keys: `param_analysis`, `observation_check`, and `questions`.

User Prompt

GENERATE TWO NATURAL BIOMEDICAL QUESTIONS

API DOCUMENTATION
(Understand the tool's specific bias and domain:)
`[API_DOC_TEXT]`

STEP 1 – Classify parameters
PARAMS:
`[PARAMS_JSON]`

STEP 2 – Check observation
(Distinguish between AVAILABLE data and MISSING/EMPTY data)
OBSERVATION:
`[OBSERVATION_JSON]`

STEP 3 – Write TWO questions

- **Question 1:** Broad intent (Natural tone, implies need for this specific tool).
- **Question 2:** Specific feature (Focus on a field that HAS data).
- **MUST include these identifiers:** `[IDENTIFIERS_BLOCK]`

REFERENCE EXAMPLES
`[FEW_SHOTS_TEXT]`

Output JSON with `param_analysis`, `observation_check`, and `questions`.

E.2 Prompt for informative check

The following prompt is used to evaluate whether an observation is informative enough to answer a specific user query. It requires (1) the natural language user query and (2) the JSON representation of the biological observation.

System Prompt

You are an evaluator for dataset filtering.

Goal: Decide whether the OBSERVATION is informative (useful) for answering the USER QUERY.

IMPORTANT CONTEXT: Observations are POST-PROCESSED SUMMARIES (often partial). This is NOT a strict completeness check.

Be concise and deterministic.

User Prompt

USER QUERY:
`[USER_QUERY_TEXT]`

OBSERVATION (tool output / retrieved data):
`[OBSERVATION]`

RUBRIC

- **informative=true** if the observation contains at least ONE relevant, non-trivial fact that can be used to answer part of the query without inventing details.
  - Examples/partial lists still count.
  - Counts/aggregates/summaries still count.
  - If the query asks "which X" but the observation only gives a count or a few examples, that is still `informative=true` (partial answer).
- **informative=false** ONLY when:
  - Observation is an error / empty / placeholder, OR
  - Observation content is clearly unrelated to the query intent, OR
  - Observation is too vague to support even a single concrete statement relevant to the query.

When writing the reason:

- Focus on what CAN be answered using the observation (even partially).
- If partial, put the missing parts into limitations, but do NOT flip to false just because it's incomplete.
- Be concise and specific (name the fields/signals you used).

OUTPUT FORMAT
(do NOT output JSON; output exactly these lines)
INFORMATIVE: `true|false`
REASON: `<short reason>`
LIMITATIONS: `<optional; if none, write "none">`

E.3 Prompt for generating answers

The following prompts are used to generate the final natural language responses for the human expert evaluation. These prompts require the original user query, the generated api call, and the corre-

sponding biological observations as input.

System Prompt (Base Model)

You are a concise, accurate biomedical assistant. Answer the user question as directly as possible in 2–5 sentences, using your general biomedical knowledge and reasonable domain assumptions. Do NOT mention tools, APIs, databases, internet access, or that you cannot look things up. Do NOT tell the user how to get the information. Answer directly. If the question asks for record-level details you cannot know exactly, give the best plausible answer in a natural, helpful way without refusals or meta statements (avoid phrasing like "I can't", "I don't know", "without risk of error").

User Prompt (Base Model)

`[USER_QUERY]`

System Prompt (Tool-Augmented)

You are a concise, accurate biomedical assistant. You are given a user question plus a tool call and its observation output. Answer in 2–6 sentences. Use the observation as primary evidence and your general knowledge. Write the answer directly as if you already know the facts. If the observation is insufficient, you may add general biomedical context, but do not invent record-level facts that should come from the observation.

User Prompt (Tool-Augmented)

User question:
`[USER_QUERY]`

API call (for context):
`[API_CALL_JSON]`

Observation (tool output):
`[OBSERVATION_TEXT]`

F Tool and API List

The following part enumerates all tools and their corresponding APIs used in this work, grouped by data source.

NCBI Tools

- **ESearch**
  - esearch
- **ELink**
  - elink
- **EFetch**
  - efetch
- **EInfo**
  - einfo
- **BLAST**
  - blast

UniProt Tools

- **UniProtKB**
  - get_uniprotkb_entry
  - search_uniprotkb
  - stream_uniprotkb
- **UniRef**
  - get_uniref_by_id
  - get_uniref_light
  - get_uniref_members
  - search_uniref
  - stream_uniref
- **UniParc**
  - get_uniparc_by_upi
  - get_uniparc_databases
  - get_uniparc_light
  - search_uniparc
  - stream_uniparc
- **GeneCentric**
  - get_genecentric_by_accession
  - get_genecentric_by_proteome_id
  - search_genecentric
  - stream_genecentric
- **Proteomes**
  - get_proteome_by_upid
  - search_proteomes
  - stream_proteomes
- **Literature citations**
  - get_citation_by_id
  - search_literature_citations
  - stream_literature_citations
- **Keywords**
  - get_keyword_by_id
  - search_keywords
  - stream_keywords
- **Human diseases**
  - get_disease_by_id
  - search_human_diseases
  - stream_human_diseases
- **Subcellular locations**
  - get_location_by_id
  - search_subcellular_locations
  - stream_subcellular_locations
- **Cross-referenced databases**
  - get_crossref_database_by_id
  - search_crossref_databases
  - stream_crossref_databases
- **Taxonomy**
  - get_taxonomy_by_id
  - search_taxonomy
  - stream_taxonomy
- **UniRule**
  - get_unirule_by_id
  - search_unirule
  - stream_unirule
- **ARBA**
  - get_arba_by_id
  - search_arba
  - stream_arba
- **Archive**
  - get_archive_id

Ensembl Tools

- **Comparative Genomics**
  - get_alignment_region
  - get_cafe_genetree_by_id
  - get_cafe_genetree_by_member_id
  - get_cafe_genetree_by_member_symbol
  - get_genetree_by_id
  - get_genetree_member_by_id
  - get_genetree_member_by_symbol
  - get_homology_by_id
  - get_homology_by_symbol
- **Cross References**
  - get_xrefs_by_id
  - get_xrefs_by_symbol
  - lookup_xref_name
- **Information**
  - get_info_analysis
  - get_info_assembly
  - get_info_assembly_region
  - get_info_biotypes
  - get_info_biotypes_groups
  - get_info_biotypes_name
  - get_info_compara_methods
  - get_info_compara_species_sets
  - get_info_external_dbs
  - get_info_genomes
  - get_info_genomes_accession
  - get_info_genomes_assembly
  - get_info_genomes_division
  - get_info_genomes_taxonomy
  - get_info_species
  - get_info_variation_population_name
  - get_info_variation_populations
  - get_info_variation_sources
- **Linkage Disequilibrium**
  - get_ld_around_variant
  - get_ld_pairwise
  - get_ld_region
- **Lookup**
  - lookup_by_id
  - lookup_by_symbol
- **Mapping**
  - map_assembly
  - map_cdna_to_genome
  - map_cds_to_genome
  - map_translation_to_genome
- **Ontologies and Taxonomy**
  - get_ontology_ancestors
  - get_ontology_ancestors_chart
  - get_ontology_descendants
  - get_ontology_id
  - get_ontology_name
  - get_taxonomy_classification
  - get_taxonomy_id
  - get_taxonomy_name
- **Overlap**
  - overlap_by_id
  - overlap_by_region
  - overlap_translation
- **Phenotype annotations**
  - get_phenotype_by_accession
  - get_phenotype_by_gene
  - get_phenotype_by_region
  - get_phenotype_by_term
- **Regulation**
  - get_binding_matrix
- **Sequence**
  - get_sequence_by_id
  - get_sequence_by_region
- **Transcript Haplotypes**
  - get_transcript_haplotypes
- **VEP**
  - vep_by_hgvs
  - vep_by_id
  - vep_by_region
- **Variation**
  - get_variation
  - get_variation_by_pmcid
  - get_variation_by_pmid
  - variant_recoder
- **Variation GA4GH**
  - get_ga4gh_callsets
  - get_ga4gh_datasets
  - get_ga4gh_features
  - get_ga4gh_featuresets
  - get_ga4gh_references
  - get_ga4gh_referencesets
  - get_ga4gh_variantannotationsets
  - get_ga4gh_variants
  - get_query_beacon