Beyond GeneGPT: A Multi-Agent Architecture with Open-Source LLMs for Enhanced Genomic Question Answering

Haodong Chen
The University of Queensland
Brisbane, Australia
haodong.chen1@student.uq.edu.au

Guido Zuccon
The University of Queensland
Brisbane, Australia
g.zuccon@uq.edu.au

Teerapong Leelanupab
The University of Queensland
Brisbane, Australia
t.leelanupab@uq.edu.au

ABSTRACT

Genomic question answering (QA), such as interpreting gene disease associations or querying protein functions, often requires complex reasoning and integration across diverse biomedical sources. GeneGPT addressed this challenge by combining domain-specific APIs with a large language model, specifically OpenAI's code-davinci-002, to enable natural language interaction with genomic databases. However, its reliance on a proprietary model limits scalability, increases operational costs, and raises concerns about data privacy and generalization.

In this work, we revisit and reproduce GeneGPT in a pilot study using open source models, including Llama 3.1, Qwen2.5, and Qwen2.5 Coder, within a monolithic architecture. Building on this foundation, we propose *OpenBioLLM*, a modular multi-agent framework that extends GeneGPT by introducing agent specialization for tool routing, query generation, and response validation. This enables coordinated reasoning and role-based task execution.

OpenBioLLM matches or outperforms GeneGPT on over 90% benchmark tasks, achieving average scores of 0.849 on GeneTuring and 0.830 on GeneHop, while using smaller open-source models without additional fine-tuning or tool-specific pretraining. Its modular multi-agent design reduces latency through parallel and lightweight task execution. The results of our comprehensive evaluation highlight the potential of open-source multi-agent systems for genomic QA.

CCS CONCEPTS

• **Information systems → Specialized information retrieval**.

KEYWORDS

LLM, Open-Source Model, Genomic Question Answering, NCBI

ACM Reference Format:
Haodong Chen, Guido Zuccon, and Teerapong Leelanupab. 2025. Beyond GeneGPT: A Multi-Agent Architecture with Open-Source LLMs for Enhanced Genomic Question Answering. In . ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION AND RELATED WORK

Large Language Models (LLMs) have achieved remarkable success in general-domain NLP tasks, with state-of-the-art closed-source models like GPT-4o [12], Claude 3.5 [1], and Gemini 1.5 [3]. Meanwhile, open-source models such as Llama 3.1 [18], Qwen 2.5 [14], and Mistral [11] are rapidly closing the performance gap, enabling greater reproducibility and customization.

Beyond general NLP, LLMs are increasingly applied to specialized domains, such as finance [19], law [23], clinical medicine [16], and biomedicine [10]. In genomics, GeneGPT [6] pioneered the integration of LLMs with domain-specific APIs from the National Center for Biotechnology Information (NCBI) [15], enabling tool-augmented question answering with improved factual accuracy.

Despite its contributions, GeneGPT is built on a monolithic pipeline using a single proprietary Codex model (code-davinci-002), which is now deprecated and unavailable to the public. This limits its reproducibility, scalability, and extensibility. Additionally, its single-agent architecture lacks the flexibility to support multi-hop reasoning and complex task decomposition, as required by benchmarks like GeneHop [6].

In this work, we revisit and reimplement GeneGPT using open-source models, including Llama 3.1, Qwen 2.5, and Qwen2.5 Coder, within its original monolithic design. Building on this foundation, we propose *OpenBioLLM*, a modular multi-agent framework that introduces agent specialization for tool routing, query generation, and response validation. This architecture supports coordinated reasoning, parallel execution, and role-based task delegation to handle diverse genomic queries more effectively.

We comprehensively evaluate OpenBioLLM on twelve benchmark tasks, showing that it matches or outperforms GeneGPT on eleven, achieving average scores of 0.849 on GeneTuring and 0.830 on GeneHop, while using smaller open-source models without additional fine-tuning or tool-specific pretraining. Moreover, its modular multi-agent design reduces latency and improves interpretability.

Beyond validating prior work [6], our study is driven by the following goals, each of which informs a key contribution:

1. **Generalizability and Reproducibility:** We adapt GeneGPT's pipeline to open-source LLMs, enabling reproduction without reliance on proprietary models.
2. **Improved Multi-Step Reasoning:** We enhance performance on complex tasks (e.g., GeneHop) by analyzing failure cases and leveraging complementary model strengths through modular design.
3. **Tool Use with Open-Source Models:** We demonstrate that open-source LLMs can support tool-augmented QA via prompt design and agent coordination without fine-tuning or task-specific supervision.

Table 1: GeneTuring Tasks, Categories, Inputs, and Outputs

GeneTuring Task	Input	Output
Nomenclature		
Gene alias	Gene alias	Official gene symbol
Gene name conversion	Ensembl Gene ID	Official gene symbol
Genomic Location		
Gene location	Gene name	Chromosomal location
SNP location	SNP ID	Chromosomal location
Gene SNP association	SNP ID	Related genes
Functional Analysis		
Gene disease association	Disease name	Related genes
Protein-coding genes	Gene name	Indicates if it's a protein-coding gene
Sequence Alignment		
DNA to human genome	DNA sequence	Chromosomal location
DNA to multiple species	DNA sequence	Chromosomal locations across species

4. **Multi-Agent Architecture:** We introduce a modular framework with specialized agents for tool routing, query generation, and response validation, enabling coordinated reasoning and flexible task execution beyond GeneGPT's monolithic design.
5. **API Dependency Analysis:** We assess the limitations of relying on domain-specific APIs (e.g., NCBI) and offer insights for improving robustness and generalization in biomedical QA systems.

Overall, our work goes beyond reproducing GeneGPT by proposing a flexible multi-agent framework that integrates refined prompt structures, API-based tool use, structured multi-step reasoning, and well-defined few-shot examples. We evaluate the effectiveness and generalizability of these components beyond genomic information retrieval. All code and resources are publicly available at https://anonymous.4open.science/r/OpenBioLLM-50EE.

2 TASK BACKGROUND

We evaluate our system on two genomic QA benchmarks: GeneTuring and GeneHop. These tasks assess the ability of LLMs to answer structured and multi-step questions using genomic data and APIs.

2.1 GeneTuring

GeneTuring [5] consists of twelve tasks (50 questions each). Following GeneGPT [6], we focus on the nine tasks that interact with NCBI APIs. These tasks fall into four functional categories, as outlined in Table 1, which summarizes the input and output for each:

Nomenclature: Standardizes gene names through alias identification and Ensembl-to-symbol conversion.
Genomic Location: Retrieves chromosomal positions of genes or SNPs, as well as their associations.
Functional Analysis: Identifies disease-associated or protein-coding genes.
Sequence Alignment: Maps DNA sequences to genomic coordinates or identifies the species (e.g., "worm," "chicken," "rat") to which the sequence belongs.

All tasks follow a single-hop QA format, with answers retrieved through structured calls to NCBI APIs (e.g., E-utils, BLAST).

Table 2: GeneHop Tasks, Inputs, Outputs, and Required APIs

GeneHop Task	Input	Output	Required APIs
SNP gene function	SNP ID	Function description and associated gene	e-utils
Disease gene location	Disease name	Chromosomal location	e-utils
Sequence gene alias	DNA sequence	Gene alias	blast + e-utils

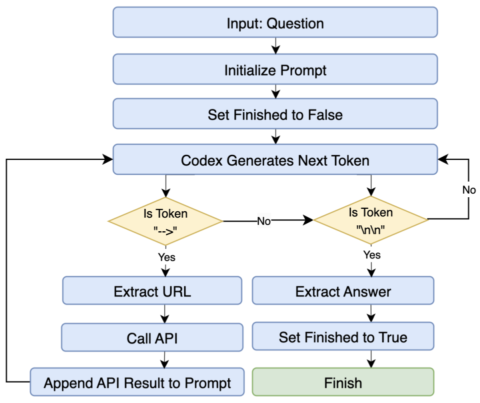

Figure 1: Inference algorithm in GeneGPT

2.2 GeneHop

GeneHop [6] evaluates a model's ability to perform multi-hop reasoning via three tasks requiring sequential API calls and intermediate logic. Each task begins with a primary question and involves decomposing the query into multiple steps.

SNP Gene Function: Identify the gene associated with a given SNP, then retrieve its function (SNP $\longrightarrow$ gene $\longrightarrow$ function).
Disease Gene Location: Retrieve the OMIM ID for a disease, identify related genes, and determine their chromosomal locations (disease $\longrightarrow$ OMIM ID $\longrightarrow$ gene $\longrightarrow$ location).
Sequence Gene Alias: From a DNA sequence, infer the genomic location, retrieve the gene, and return its aliases (sequence $\longrightarrow$ location $\longrightarrow$ gene $\longrightarrow$ aliases).

These tasks simulate real-world genomic queries and test a model's ability to integrate multiple API responses while maintaining reasoning consistency. Table 2 summarizes task inputs, outputs, and associated NCBI APIs.

3 GENEGPT

3.1 GeneGPT Inference and NCBI APIs

GeneGPT [6] enables LLMs to interact with biomedical databases via NCBI Web APIs [15], combining structured prompts with tool-augmented generation to answer genomic questions.

The prompt comprises four components: a task description, API documentation, example demonstrations, and a test question. GeneGPT was introduced in two versions: a **-full** version, which includes two API documentation blocks and four demonstrations, and a **-slim** version, which omits documentation and includes only two demonstrations. For GeneHop, the prompt was further tailored per task, depending on whether information retrieval required only E-utils or both E-utils and BLAST (see Table 2). These components guide the inference process, as illustrated in Figure 1. Table 3 summarizes the APIs and example usages embedded in the prompt.

3.2 The Irreproducibility of GeneGPT

Although GeneGPT achieved strong performance by prompting Codex to interact with NCBI APIs, it is no longer reproducible. The Codex models used in its pipeline were officially shut down in 2024 and are no longer publicly available.[*]

[*]https://platform.openai.com/docs/deprecations#instructgpt-models

Table 3: Summary of NCBI API usage documentations and demonstrations in the GeneGPT prompt.

Comp.	Doc/Demo	Database	Function	GeneTuring		GeneHop†	
				-full	-slim	e-utils	blast + eutils
Doc.1	E-utils	gene, snp, omim	esearch, efetch, esummary	✓	○	✓	✓
Doc.2	BLAST	nt	blastn	✓	○	○	✓
Demo.1	Alias	gene	esearch → efetch	✓	✓	✓	✓
Demo.2	Gene SNP	snp	esummary	✓	○	✓	○
Demo.3	Gene disease	omim	esearch → esummary	✓	○	✓	✓
Demo.4	Alignment	nt	blastn	✓	✓	○	✓

†The results of GeneGPT and OpenBioLLM-based models on GeneHop will be aggregated and reported as the overall average across all tasks, regardless of version.

This limitation highlights a broader concern: reliance on proprietary models risks future irreproducibility once those models are deprecated. To address this, we reproduce GeneGPT using open-source models, ensuring that the system remains accessible, reproducible, and sustainable for future research in genomic QA.

4 PILOT STUDY: REPRODUCING GENEGPT

4.1 Open-source LLMs with GeneGPT

To assess the feasibility of open-source LLMs for genomic QA, we evaluated mainstream open-source models with strong multi-hop reasoning, RAG, and tool-use capabilities. Based on these criteria, we selected Llama3.1, Qwen2.5, and Qwen2.5-Coder.

Llama3.1 has been widely applied across various domains, particularly in healthcare and biomedical research. Prior have shown that Llama3.1-70B improves accuracy in medical QA, pharmacogenomic data extraction, and biomedical research, making it a strong candidate for handling genomic queries [8, 17, 20]. Additionally, its effectiveness in multi-hop reasoning tasks [4] aligns well with GeneHop, reinforcing its suitability for complex information retrieval and synthesis.

Qwen2.5 has demonstrated superior multi-hop reasoning and tool-use capabilities compared to Llama3.1, offering greater accuracy and stability with a lower error rate [22]. It also outperforms other open-source models in financial misinformation detection[13] and low-resource language understanding [7]. Given that genomic datasets often require retrieval from structured sources, such as NCBI, Qwen2.5's proficiency in RAG makes it particularly well-suited for our study.

Qwen2.5-Coder extends Qwen2.5's capabilities with enhanced code generation and structured data processing, making it more effective for API-based tool use and structured retrieval tasks. Since GeneGPT relies on precise API calls to retrieve genomic information from NCBI, a model optimized for coding tasks and structured retrieval is expected to improve performance. Additionally, prior studies indicate that Qwen2.5-Coder exhibits strong generalization in zero-shot and few-shot settings [7], which is crucial for handling genomic questions.

4.2 Selecting Model Sizes for Evaluation

To evaluate the impact of model size on achieving GeneGPT-like performance, we conducted an initial test using Llama3.1 with sizes of 8B and 70B. The results indicated that the 8B model was unable to comprehend tasks of this nature, while the 70B model produced performance more aligned with the expected outcomes. Based on these findings, we established 70B as the baseline model size.

For Qwen2.5, we selected the 72B version to maintain consistency with this baseline for comparison. In contrast, Qwen2.5-Coder, designed specifically for coding tasks, is only available in a 32B configuration. Despite its smaller size, we included Qwen2.5-Coder-32B in the experiment to assess its specialized capabilities and potential applicability to the task. For simplicity, we refer to Llama3.1-70B as Llama, Qwen2.5-72B as Qwen, and Qwen2.5-Coder-32B as Qwen-Coder throughout the study until further optimization.

4.3 Experimenting with GeneGPT Prompt

For the GeneTuring, we initially adopted the same standard prompting strategy ("-full") and pipeline as GeneGPT, replacing Codex (code-davinci-002) with the three selected open-source LLMs for evaluation. We then experimented the best performing open-source LLMs with a compact prompt "-slim", including only two demonstrations (Table 3). The experiments were conducted on Google Cloud, utilizing an accelerator-optimized compute engine with with an A2 standard instance (a2-highgpu-2g), equipped with 2 NVIDIA A100 GPUs (80 GB GPU memory), 24 vCPUs, and 170 GB instance memory. Model deployment was managed via Ollama [2], while LlamaIndex [9] was used to access the Ollama LLM client locally.

4.4 Evaluation methods

To ensure the consistency and reproducibility of the evaluation, we wrote a set of automated scripts to batch analyze and score the model outputs of 9 tasks in the GeneTuring dataset and 3 tasks in the GeneHop dataset. The overall evaluation criteria remain consistent with the original GeneGPT paper, as shown in Table 4.

However, several tasks involve a certain degree of semantic understanding and subjective judgment, and they cannot be accomplished by simply relying on string matching or recall-based approaches. These include SNP gene function, Multi-species DNA alignment, and Protein-coding genes, for which we introduce a large language model as an evaluator—specifically, the Qwen2.5 32B model—to perform semantic-level scoring. The model outputs a score of 1, 0.5, or 0 based on the semantic consistency between the candidate answer and the reference answer, which is a more reasonable reflection of the quality of responses in complex tasks.

Table 4: Evaluation Criteria for GeneTuring and GeneHop

Task	Evaluation Criteria
GeneTuring Tasks	
Gene alias	String match
Gene disease association	Recall (correct answers / ground-truth answers)
Gene location	String match
Human genome DNA alignment	Exact match=1 point; correct chromosome but incorrect position=0.5 point
Multi-species DNA alignment	String match (species name mapping)
Gene name conversion	String match
Protein-coding genes	String match (binary mapping of correctness)
Gene SNP association	String match
SNP location	String match
GeneHop Tasks	
SNP gene function	Manual semantic judgement (1, 0.5, or 0 point)
Disease gene location	Recall (correct answers / ground-truth answers)
Sequence gene alias	Recall (correct answers / ground-truth answers)

4.5 Open LLM Results using GeneGPT pipeline

Sample-Based Evaluation: Before conducting a full-scale assessment of open LLMs with the GeneGPT pipeline, we performed a pilot evaluation by sampling the first ten questions (10Qs) from the 50 available per task in both GeneTuring and GeneHop. This approach allowed us to assess preliminary model performance before proceeding with optimization and comprehensive evaluation.

Results in GeneTuring: Table 5 shows that *Llama-full* significantly underperformed compared to GeneGPT-full, with an average accuracy of only 0.19 versus 0.84. For example, it scored just 0.20 in Gene alias and 0.10 in gene-disease association, where GeneGPT-full achieved near-perfect results.

Table 5: Performance comparison of three models (first ten GeneTuring questions per task).

GeneTuring Task (10Qs)	Llama-full	Qwen-full	Qwen-Coder-full	GeneGPT-full
Gene alias	0.20	**1.00**	0.30	**1.00**
Gene disease association	0.10	**1.00**	0.25	0.85
Gene location	0.00	**0.70**	0.00	**0.70**
Human genome DNA alignment	0.20	**0.50**	**0.50**	**0.50**
Multi-species DNA alignment	0.80	**0.90**	**0.90**	**0.90**
Gene name conversion	0.00	**1.00**	0.10	**1.00**
Protein-coding genes	0.10	**1.00**	0.50	0.60
Gene SNP association	0.30	**1.00**	0.20	**1.00**
SNP location	0.00	0.90	0.10	**1.00**
Average	0.19	**0.89**	0.32	0.84

In contrast, *Qwen-full 72B* outperformed GeneGPT-full across most tasks, achieving an average accuracy of 0.89. It obtained perfect scores in five tasks and notably outperformed GeneGPT-full in gene-disease association (1.00 vs. 0.85) and protein-coding genes (1.00 vs. 0.60), while performing comparably in gene location and human genome DNA alignment.

We also evaluated *Qwen-Coder*, a code-focused variant of Qwen2.5, which performed poorly overall. While it could construct API request links, it failed to extract meaningful answers from responses, highlighting a key limitation for code-specialized models in knowledge extraction tasks.

Overall, these pilot evaluations suggest that while Llama struggled with the GeneGPT pipeline, Qwen demonstrates strong potential as a replacement model. However, as these results are based on only ten questions per task, full-scale evaluation is required to confirm generalizability.

Nonetheless, this evaluation was based on only ten questions per task, which may not fully reflect performance across a broader range of scenarios. Further testing is required to confirm its generalizability and reliability.

Results in GeneHop: The pilot evaluation on GeneHop showed that GeneGPT achieved a moderate overall score of 0.62 (see Table 6). While this aligns with the original GeneGPT study [6] (0.50 on full evaluation of 50 questions, outperforming the baseline model New Bing at 0.24), its performance remains suboptimal, leaving room for improvement.

In contrast, Qwen outperformed GeneGPT across all tasks, scoring 1.00, 0.80, and 0.67 compared to GeneGPT's 0.85, 0.70, and 0.30, respectively. This underscores Qwen's superior multi-step reasoning capabilities, making its integration into the OpenBioLLM framework a promising avenue for further exploration.

Table 6: Performance Comparison: Qwen vs GeneGPT on GeneHop (first ten questions per task).

GeneHop Task (10Qs)	Qwen			Score	
	Correct	Half	Error	Qwen	GeneGPT
SNP gene function	10	0	0	**1.00**	0.85
Disease gene location	7	2	1	**0.80**	0.70
Sequence gene alias	5	3	2	**0.67**	0.30
Average	-	-	-	**0.82**	0.62

5 DETAILED OBSERVATION (OBS.) ON QWEN

Qwen has demonstrated great potential in handling tasks similar to those explored in the original GeneGPT study. To build on this progress, we undertook a detailed error analysis of the reproduction experiments. By manually examining model outputs and investigating the sources of incorrect answers, we identified several recurring failure patterns that point to specific opportunities for refinement.

5.1 Obs. 1: Exact Match & Case Sensitivity

Some errors arise from the benchmark's strict requirement for exact matches of gene symbols, which conflicts with the model's behavior under the original GeneGPT prompt that handles gene symbols in a case-insensitive manner. For example, in response to the question "*What is the official gene symbol of GalNAc-T4?*", the model returned "GALNT4", which is an alias corresponding to the case-insensitive match "GALNAC-T4". However, the ground-truth answer is "POC1B-GALNT4", which is the official gene symbol linked to the alias "GalNAc-T4" with correct casing. Because the model ignored case distinctions, it selected an alias with a similar surface form but failed to resolve it to the exact official symbol required by the benchmark.

5.2 Obs. 2: Relevant Answer Ranked Too Low

Another common issue occurs because the relevant answer appears too far down in the response list. Since HTTP requests retrieve only the top 5 or 10 items, it may be excluded entirely. These errors stem from limitations in both the model and the NCBI database.

A concrete example is the question: "*Which chromosome is the RGS16 gene located on in the human genome?*", the LLM omitted the "sort=relevance" parameter in its HTTP request. This omission has little effect on small datasets but causes irrelevant results to be ranked higher in large datasets. Unlike Qwen-Coder, the Qwen 72B model rarely overlooks this parameter. Emphasizing its inclusion in HTTP requests is crucial to mitigating this issue.

5.3 Obs. 3: Query Difficulty

Some mistakes can be attributed to ambiguous query interpretation and inconsistent API result ordering. For example, in response to "*What is the official gene symbol of TFA?*", the model generated a correct HTTP request, but the response was challenging. TFA is both an official symbol and an alias. Here, the question treats it as an alias, but the API prioritizes the official symbol, pushing the alias reference lower in the list—leading the model to select the wrong answer.

5.4 Obs. 4: Discrepancies in Coordinate Mapping Between Genome Assemblies

In the DNA alignment task, while GeneGPT and Qwen in OpenBioLLM consistently identify the correct chromosome, their inferred genomic coordinates often deviate from the reference standard. This discrepancy arises because the GeneTuring benchmark defines ground-truth coordinates using the reference assembly of `BSgenome.Hsapiens.UCSC.hg38`, whereas queries to NCBI's `core_nt` database frequently return results based on other assemblies (e.g., GRCh37 or clone-based sequences). As a result, while chromosome predictions are accurate, their exact positions may not align with the hg38 standard.

6 PROMPT ENGINEERING FOR QWEN2.5

Building on our previous study, where Qwen-72B outperformed GeneGPT, LLaMA-70B, and Qwen-Coder-7B in a 10-sample-based evaluation, we now focus on optimizing Qwen by refining prompts. Overall, we do the following prompt improvement:

1. Emphasized the output format: To improve response consistency, we explicitly instructed the model to follow a standardized output format:

1. When you need to make an API call, please put the URL you generated directly in []. Do not add anything else.
2. When you reach the final answer, please use the following format: `Answer: your generated answer.`

Enforcing this structure ensured accurate, consistent responses while maintaining adherence to the expected format.

2. Correct the missing parameters: A significant portion of errors stemmed from queries that failed to retrieve the most relevant results. This was due in part to an inconsistency in the original prompt design: while the demonstration examples included the "sort=relevance parameter," the accompanying documentation omitted it. This mismatch created ambiguity, especially for smaller models (e.g., 14B and 32B), making it unclear when the parameter should be used. To address this, we updated the documentation to explicitly include "sort=relevance" for consistency and reliability.

Additionally, we added the missing "retmode=json" parameter to the documentation. By default, the NCBI API returns responses in XML format, which poses two main issues: JSON is easier for LLMs to parse accurately, and XML responses are longer, increasing token usage. For instance, a gene esummary API query returns **5,659** tokens in JSON but **6,820** in XML. Switching to JSON reduces token consumption, lowers processing overhead, and allows more information to fit within the model's context window.

3. Incorporating ReAct-style Prompting: To better tackle multi-hop reasoning tasks in GeneHop, we draw inspiration from the ReAct prompting framework [21], which interleaves *reasoning* steps with *actions*. In ReAct, large language models (LLMs) are prompted to think step-by-step ("Thought"), execute external tool calls ("Action"), observe results ("Observation"), and make iterative decisions accordingly. This approach improves transparency, control, and performance in complex workflows.

Building on this principle, we design domain-specific ReAct-style prompts to explicitly guide the model through sub-question decomposition and tool interaction. Unlike vanilla prompting, our design not only teaches the model how to decompose complex biomedical queries, but also instructs it on what to do after receiving a tool response—e.g., which field to extract or what the next reasoning step should be. An example is shown in Appendix A.1

6.1 Optimization Test on GeneTuring

After refining the prompt, we evaluated it on the GeneTuring benchmark by running all nine tasks (450 questions in total) with Qwen-32B and Qwen-72B using the full set of API documentations and demonstrations. Both models achieved a macro-average accuracy of 0.838, which was the highest among the settings tested at this stage. These results demonstrate the effectiveness of our optimized prompt design. Detailed results are presented in Table 7[†].

[†]GeneGPT results are taken from the original publication and were not re-run in this study, as Codex (code-davinci-002) has been deprecated.

Table 7: Optimized performance on GeneTuring tasks: Qwen-32B-full and -72B-full vs GeneGPT variants.

GeneTuring Task	Qwen-full						Score			
	Correct		Half		Errors		Qwen-full		GeneGPT[†]	
	32B	72B	32B	72B	32B	72B	32B	72B	full	slim
Gene alias	45	44	0	0	5	6	**0.90**	0.88	0.80	0.84
Gene disease association	34	34	5	3	11	13	0.73	0.71	**0.76**	0.70
Gene location	35	36	0	0	15	14	0.70	**0.72**	0.62	0.66
Human genome DNA alignment	0	0	45	45	5	5	**0.45**	**0.45**	0.44	0.44
Multi-species DNA alignment	41	40	0	0	9	10	0.82	0.80	0.88	**0.90**
Gene name conversion	50	50	0	0	0	0	**1.00**	**1.00**	**1.00**	**1.00**
Protein-coding genes	50	49	0	0	0	1	**1.00**	0.98	0.74	0.98
Gene SNP association	50	50	0	0	0	0	**1.00**	**1.00**	**1.00**	**1.00**
SNP location	47	50	0	0	3	0	0.94	**1.00**	**1.00**	0.98
Average	-	-	-	-	-	-	**0.838**	**0.838**	0.804	0.833

This result shows that prompt engineering significantly improves LLM performance in genomic QA, allowing open-source models to match a proprietary one in effectiveness.

6.2 Optimization Test on GeneHop

Table 8[†] summarizes Qwen's optimized performance on GeneHop tasks. Qwen-32B and 72B demonstrated substantial improvements over GeneGPT, particularly in the sequence gene alias task, where scores increased from 0.35 to 0.75 (32B) and 0.71 (72B). In SNP gene function, Qwen-72B outperformed both Qwen-32B and GeneGPT, achieving the highest score of 0.93. However, in disease gene location, Qwen-32B struggled significantly, with its score dropping to 0.17, while Qwen-72B achieved 0.54, still lower than GeneGPT's 0.67. On average, Qwen-72B outperformed both Qwen-32B and GeneGPT, achieving a 0.729 overall score, highlighting the benefits of a larger model size for GeneHop tasks.

Table 8: Performance on GeneHop tasks after applying "ReAct-style prompt": Qwen-32B and -72B vs GeneGPT.

GeneHop Task	Qwen						Score		
	Correct		Half		Error		Qwen		GeneGPT[†]
	32B	72B	32B	72B	32B	72B	32B	72B	
SNP gene function	40	45	7	3	3	2	0.87	**0.93**	0.61
Disease gene location	8	24	1	6	41	20	0.17	0.54	**0.67**
Sequence gene alias	34	16	8	27	8	7	**0.75**	0.71	0.35
Average	-	-	-	-	-	-	0.597	**0.729**	0.543

6.3 Ablation Study

GeneGPT's prompt structure is summarized in Table 3. In their ablation study, GeneGPT demonstrated that a streamlined prompt containing only two demonstrations (referred to as the slim version) generalized well across tasks, sometimes even outperforming the full version.

We adopted a similar slim prompt setup for Qwen-32B and -72B, including only two demonstrations: gene alias resolution and DNA alignment. While Qwen-72B-slim achieved performance comparable to its full version, Qwen-32B-slim saw a significant drop, from 0.838 to 0.467 (Table 9[†]). This suggests that larger models exhibit stronger cross-task generalization under prompt compression.

Table 9: Ablation Performance on GeneTuring tasks: Qwen vs GeneGPT variants.

GeneTuring Task	Qwen-slim						Score					
	Correct		Half		Errors		Qwen				GeneGPT[†]	
	32B	72B	32B	72B	32B	72B	32B full	32B slim	72B full	72B slim	full	slim
Gene alias	37	43	0	0	13	7	**0.90**	0.74	0.88	0.86	0.80	0.84
Gene disease association	31	36	6	2	13	12	0.73	0.69	0.71	0.75	**0.76**	0.70
Gene location	7	28	0	0	43	22	0.70	0.14	**0.72**	0.56	0.62	0.66
Human genome DNA align.	0	0	39	47	11	3	0.45	0.39	0.45	**0.47**	0.44	0.44
Multi-species DNA align.	40	38	0	0	10	12	0.82	0.80	0.80	0.76	0.88	**0.90**
Gene name conversion	0	49	0	0	50	1	**1.00**	0.00	**1.00**	0.98	**1.00**	**1.00**
Protein-coding genes	45	48	0	0	5	2	**1.00**	0.90	0.98	0.96	0.74	0.98
Gene SNP association	16	50	0	0	34	0	**1.00**	0.32	**1.00**	**1.00**	**1.00**	**1.00**
SNP location	11	49	0	0	39	1	0.94	0.22	**1.00**	0.98	**1.00**	0.98
Average	-	-	-	-	-	-	**0.838**	0.467	**0.838**	0.812	0.804	0.833

7 MULTI-AGENT ARCHITECTURE

7.1 Why Multi-Agent Architecture?

Although prompt engineering has improved performance on GeneTuring and GeneHop, the single-model architecture still faces fundamental limitations.

First, multi-step tasks require inserting all API results, including large JSON outputs, directly into the prompt. This greatly increases the context size, making it harder for the model to focus on the most relevant information, even with a large context window.

Second, assigning all responsibilities—from building API calls to extracting information and generating answers—to one model is inefficient. Different tasks benefit from different capabilities: code generation models are better at constructing requests, summarization models at extracting key content, and biomedical models at evaluating scientific accuracy. Dividing responsibilities across specialized modules improves both effectiveness and precision.

Finally, single-model systems are difficult to interpret and debug. When the answer is wrong, it is unclear whether the issue lies in the API call, data retrieval, or final reasoning step. This lack of transparency limits our ability to diagnose and improve the system.

7.2 Modular Multi-Agent Pipeline

To address the limitations of a monolithic design, we propose a modular multi-agent architecture, *OpenBioLLM*, for genomic QA. Rather than relying on a single large model to handle all stages of reasoning, our approach decomposes the workflow into specialized agents, each responsible for a distinct task such as query generation, information retrieval, summarization, or answer validation. This modularization enables more efficient use of model capabilities and improves both interpretability and traceability by exposing intermediate reasoning steps.

The pipeline is implemented using the LangGraph framework, which supports flexible control flow and message exchange between agents. The pipeline consists of six core components: three pipeline controllers—*Router*, *Evaluator*, and *Generator*—and three tool agents—*Eutils Agent*, *BLAST Agent*, and *Web Search Agent*.

The process begins with a user query, which is routed by the *Router* to the appropriate Tool Agent based on the query type. The selected Tool Agent retrieves relevant information and sends it to the *Evaluator*, which determines whether the information is sufficient to answer the question. If not, the query is returned to the Router for further processing. Once sufficient evidence has been gathered, the *Generator* synthesizes the final answer.

An overview of the architecture is provided in Figure 2. The following sections detail the roles of each component, starting with the three pipeline controllers, followed by the three tool agents.

Router Module: The Router determines which Agent to invoke based on the user question, dialog history, and Evaluator feedback. It routes requests to one of three Agents: Eutils, Blast, or Web Search.

If a question is unrelated to bioinformatics, the Router generates a polite rejection to maintain system professionalism and stability. The prompt used by this module is shown in Appendix A.2.

Evaluator Module: The Evaluator makes decisions based on the user question, conversation history, and the outputs of invoked API tools.

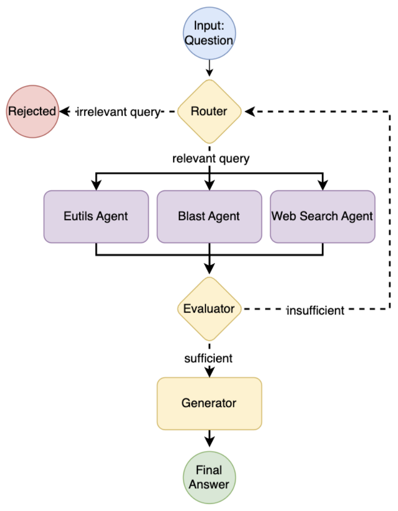

Figure 2: Multi-Agent Pipeline in OpenBioLLM Framework.

To improve interpretability and stability, its output follows a strict JSON format with two fields: `next_step` (the decided action) and `reason` (the explanation). This explicit reasoning enhances transparency, traceability, and encourages the model to reflect before proceeding. The prompt used by the Evaluator is shown in Appendix A.3.

Generator Module: In the Generator module, the system generates the final answer by combining the user query with results from Eutils, BLAST, and Web Search.

To ensure professionalism and clarity, the model is prompted to act as a bioinformatician, providing clear and concise answers that cite specific facts (e.g., gene names, sequences) and acknowledge any limitations when information is incomplete. The prompt used by this module is shown in Appendix A.4.

Eutils Agent Module: In the Eutils Agent module, the system retrieves information through a two-step process: it first obtains relevant IDs and then fetches detailed records based on those IDs.

Unlike the original GeneGPT design, which required generating full HTTP URLs, our approach outputs structured JSON parameter objects. This reduces token usage, improves traceability by recording parameter combinations, and avoids redundant requests. Hard-coded defaults (e.g., `retmode=json`, `sort=relevance`) ensure essential arguments are always included. The agent also skips requests with previously used parameters to enhance efficiency. The prompt used by this module is shown in Appendix A.5.

BLAST Agent Module: The BLAST Agent compares user-supplied DNA sequences with the reference genome through a two-step process: first, it submits the sequence via a PUT request to obtain a request ID (RID); then, after about 10 seconds, it polls the result with a GET request using the RID.

To improve robustness and efficiency, the agent records processed sequences to avoid duplicate submissions and includes wait-and-retry logic, polling every 10 seconds and retrying up to three times if results are not ready.

In the prompt design, the model is guided to extract and validate DNA sequences from user input, outputting them in strict

JSON format. Only outputs containing a valid "sequence" field are accepted. Appendix A.6 shows the detailed prompt.

Web Search Agent Module: The Web Search Agent uses the Google Custom Search API to retrieve real-time information for queries that cannot be answered by structured databases like Eutils or BLAST. It serves as a fallback for out-of-domain questions, ambiguous inputs, and unstructured queries such as general biological facts or conceptual topics.

Unlike other agents, it operates as a purely functional component: it formulates a search request based on the user question and retrieves the top 5 results, extracting key fields such as title, source, date, author, summary, and keywords. This information is formatted (Appendix A.7) and passed as a ToolMessage to the Evaluator for further processing.

7.3 Result Comparison

To evaluate the effectiveness and efficiency of our Multi-Agent Architecture, we tested different model sizes and configurations across system modules. Both the Pipeline Controller components (Router, Evaluator, Generator) and Tool Agents (Eutils, BLAST, Web Search) rely on LLMs, but their roles differ: Tool Agents mainly extract parameters and call APIs, while the Pipeline Controller manages system logic, tool selection, and answer generation.

We explored the trade-off between performance and computational cost with four configurations: (1) a lightweight setup using 14B for the Controller and 7B for Agents, (2) a balanced setup with 14B for both, (3) an enhanced setup with 32B for the Controller and 14B for Agents, and (4) a fully scaled setup using 32B models throughout. Evaluation results on GeneTuring and GeneHop are summarized in Table 10.

Our results reveal several insightful trends, described below.

1. Smaller models can achieve competitive performance: Surprisingly, using 14B models for both the Pipeline Controller and Tool Agents already yields strong results, slightly outperforming the original GeneGPT's single 72B model (average GeneTuring score: 0.849 vs. 0.838). This highlights the advantage of modular decomposition: assigning specialized tasks to moderate-sized models can surpass a monolithic LLM with far more parameters.

2. Larger controllers greatly improve GeneHop performance: Using a 32B model for the Pipeline Controller while keeping Tool Agents at 14B leads to substantial gains on the more complex GeneHop benchmark, raising the average score to 0.830. We attribute this improvement to the controller's enhanced ability to handle multi-step reasoning, assess sufficiency, and manage fallback logic. As GeneHop tasks demand deeper inference and multi-hop coordination, a stronger controller better orchestrates the multi-agent workflow.

3. Larger Tool Agents do not yield additional benefits: Unexpectedly, upgrading both the controller and Tool Agents to 32B models does not improve performance. GeneTuring results plateau at 0.848, while GeneHop scores slightly decrease. This suggests diminishing returns from scaling Tool Agents, whose tasks are relatively simple and do not require extensive reasoning. It also raises potential issues of redundant computation and increased latency due to unnecessarily large submodules.

4. Optimal trade-off: 32B Controller with 14B Tools: Overall, the best balance between performance and efficiency is achieved with a 32B model for the Pipeline Controller and 14B models for the Tool Agents. This setup delivers the highest average performance across both GeneTuring and GeneHop benchmarks while keeping resource usage reasonable. It underscores the importance of allocating model capacity based on task complexity: using larger models for reasoning and coordination, and smaller models for structured extraction or simple lookups.

7.4 Error Analysis (32B+14B Multi-Agent)

We manually analyzed errors from the GeneTuring and GeneHop benchmarks under the 32B+14B configuration. Table 11 summarizes the results. Errors were grouped into six categories (E1–E5 and "Other"), each reflecting a specific issue in reasoning or tool usage. Below, we describe each error type and note where it most commonly occurred.

E1: Incorrect API Selection: These errors occur when the model selects the wrong API for the given task. For example, it may call the BLAST interface when E-utilities is required, or vice versa.

E2: Incorrect Parameter Usage: The model selects the correct API but fails due to incorrect parameters—for example, using the wrong search term, confusing databases like `db=gene` and `db=omim`, or misusing `efetch` and `esummary`. This error is especially common in GeneHop, with 15 cases in the SNP gene function task and several in Disease gene location.

E3: Misunderstanding Task Semantics: These errors occur when the model misinterprets the question's intent despite having access to the correct information. For example, in the Gene alias task, it may treat the alias as the official name and return it directly, leading to an incorrect answer.

E4: Insufficient Information to Answer: This is the most common error type across both benchmarks. It occurs when the model follows the correct procedure and uses the right tools, but the retrieved data lacks sufficient detail to produce a valid answer.

This issue is particularly prevalent in tasks like Human genome DNA alignment and Gene location in GeneTuring, and Disease gene location in GeneHop.

Notably, adding a Web Search Agent only partially alleviates this problem. For example, for sequence-based questions such as:

Which organism does the DNA sequence come from: AAAAATAATTTCCCGTTAACTGTTAATAAGTATTAGCAG...

even searching the full sequence on Google yields little useful information. This highlights a limitation in the biomedical domain: some queries are simply not "searchable," rendering external retrieval ineffective in these cases.

E5: Incomplete Retrieval or Premature Termination: These errors occur when the model ends its reasoning too early, leading to incomplete answers. For example, in the Sequence gene alias task, it may retrieve some aliases in the first step but fail to issue additional queries for the full list, resulting in partial results.

O: Other Errors (Scoring and Evaluation Issues): This category includes errors arising from evaluation rather than reasoning. Common cases are: (1) the model retrieves multiple correct aliases but

Table 10: Performance comparison on GeneTuring and GeneHop tasks

Model	Original-GeneGPT[†] (code-davinci-002)		Optmized-GeneGPT (Qwen2.5)				OpenBioLLM[‡] (Qwen2.5)			
	GeneGPT-full	**GeneGPT-slim**	**Qwen32b-full**	**Qwen32b-slim**	**Qwen72b-full**	**Qwen72b-slim**	**Multi-(14b+7b)**	**Multi-(14b+14b)**	**Multi-(32b+14b)**	**Multi-(32b+32b)**
GeneTuring Tasks										
Gene alias	0.80	0.84	0.90	0.74	0.88	0.86	0.84	0.90	0.92	**0.96**
Gene disease association	0.76	0.70	0.73	0.69	0.71	0.75	0.48	**0.81**	0.76	0.73
Gene location	0.62	0.66	0.70	0.14	**0.72**	0.56	0.68	0.66	0.66	0.66
Human genome DNA alignment	0.44	0.44	0.45	0.09	0.45	0.47	0.47	0.47	0.46	**0.48**
Multi-species DNA alignment	0.88	**0.90**	0.82	0.80	0.80	0.76	0.84	0.84	0.82	0.82
Gene name conversion	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**
Protein-coding genes	0.74	0.71	0.90	0.90	0.98	0.96	**1.00**	0.98	**1.00**	**1.00**
Gene SNP association	**1.00**	**1.00**	**1.00**	0.32	**1.00**	**1.00**	0.94	**1.00**	**1.00**	**1.00**
SNP location	**1.00**	0.94	0.22	**1.00**	0.98	0.98	0.98	0.98	0.98	0.98
Average	0.804	0.833	0.838	0.467	0.838	0.813	0.799	**0.849**	0.844	0.848
GeneHop Tasks										
SNP gene function	0.61		0.87		**0.93**		0.63	0.82	0.85	0.78
Disease gene location	0.67		0.17		0.54		0.22	0.75	**0.79**	0.69
Sequence gene alias	0.35		0.75		0.85		0.72	0.72	0.85	**0.91**
Average	0.543		0.597		0.727		0.431	0.761	**0.830**	0.793

[†] Original GeneGPT results are taken from the published paper for comparison with our methods, including Qwen2.5 using the same monolithic pipeline with optimized prompt, and our multi-agent OpenBioLLM.
[‡] the first value indicates the Pipeline Controller's model size, while the second value indicates the Tool Agents' model size.(e.g., '14b+7b' means 14B Pipeline Controller and 7B Tool Agents)

Table 11: Error Analysis on Multi-Agent Qwen2.5 (32B + 14B)

GeneTuring Task	E1	E2	E3	E4	E5	O
Gene alias	0	0	2	2	0	0
Gene disease association	0	0	0	7	0	8
Gene location	0	0	0	17	0	0
Human genome DNA alignment	0	0	0	49	0	1
Multi-species DNA alignment	0	0	0	5	0	4
Gene name conversion	0	0	0	0	0	0
Protein-coding genes	0	0	0	0	0	0
Gene SNP association	0	0	0	0	0	0
SNP location	0	0	0	1	0	0
GeneHop Task	**E1**	**E2**	**E3**	**E4**	**E5**	**O**
SNP gene function	0	15	0	0	0	0
Disease gene location	0	7	0	5	0	1
Sequence gene alias	1	4	0	3	2	0

E1: Wrong API E2: Wrong arguments E3: Wrong comprehension
E4: Unanswerable with API E5: Insufficient query O: Others

omits a few, leading to partial correct; and (2) minor formatting differences or string mismatches cause correct answers to be marked as incorrect.

Overall, we observe that E4 (insufficient information) and E2 (parameter errors, particularly in GeneHop) are the main bottlenecks in the current system:

1. **E4 errors** are common across GeneTuring tasks, especially in Human genome DNA alignment and Gene location. These reflect a fundamental limitation: even with correct reasoning and tool use, the available data may lack sufficient coverage to fully answer the question, indicating an external bottleneck in database quality and scope.
2. **E2 errors** in GeneHop reveal the model's difficulty with flexible, loosely structured queries that require precise API parameterization. While some failures stem from the zero-shot nature of prompts, ultimately, fine-tuning for better API understanding and parameter selection would provide a more reliable solution.

7.5 Why 32B Tool Agents Underperform vs. 14B

In GeneHop, most errors concentrated in two subtasks: SNP gene function and Disease gene location. These errors were not due to limitations in model capability, but rather differences in reasoning strategies across model sizes.

For Disease gene location, the standard approach involves three steps: (1) retrieving the OMIM ID from the disease name, (2) obtaining related genes, and (3) querying their chromosome locations. The 32B model often attempts shortcuts by directly calling the OMIM esummary to extract gene and location data in one step. While faster, this yields incomplete or inaccurate results. In contrast, the 14B model follows the full pipeline, producing more reliable answers despite longer execution.

A similar pattern appears in SNP gene function tasks. Instead of completing the two-step process (retrieve gene then its function), the 32B model often stops early with vague results, leading to partial scores. The 14B model, unable to find sufficient information in the first step, proceeds to the second, ultimately providing a more complete answer.

These findings highlight that stronger models can underperform when they take overly aggressive shortcuts or terminate reasoning prematurely—illustrating a "too smart for its own good" failure mode. This underscores the need for task-specific fine-tuning and stricter control over reasoning chains to ensure consistent performance.

Moreover, we note that GeneGPT's reliance solely on NCBI tools imposes limitations, such as restricted query parameters and lack of relevance-based ranking. For instance, 17% of GeneTuring's human genome DNA alignment questions remain unanswerable with NCBI alone. Integrating additional data sources and retrieval techniques like query rewriting or expansion could improve coverage and answer quality.

8 MORE THAN ACCURACY

8.1 Latency Analysis

To further investigate the system-level benefits of our multi-agent design, we compared response times between the monolithic Qwen2.5-32B model and our multi-agent OpenBioLLM framework configured with the same 32B base model for the controller and tool agents (32B+32B). This isolates the impact of architectural differences from that of model capacity. The results are shown in Figures 3 and 4, which report per-task latency across GeneTuring and GeneHop benchmarks, respectively.

Across both benchmarks, our multi-agent architecture consistently outperforms the monolithic baseline in terms of response time. In the GeneTuring dataset (Figure 3), significant latency reductions are observed in tasks such as Gene alias, Gene location, and

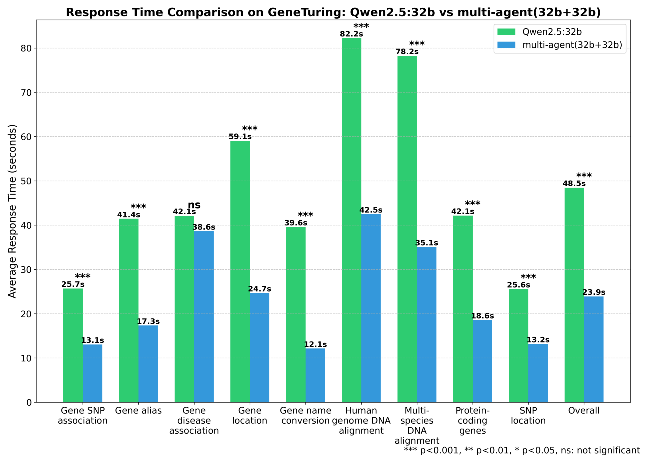

Figure 3: Task-wise latency comparison on GeneTuring: Qwen2.5-32B vs. Multi-Agent (32B+32B).

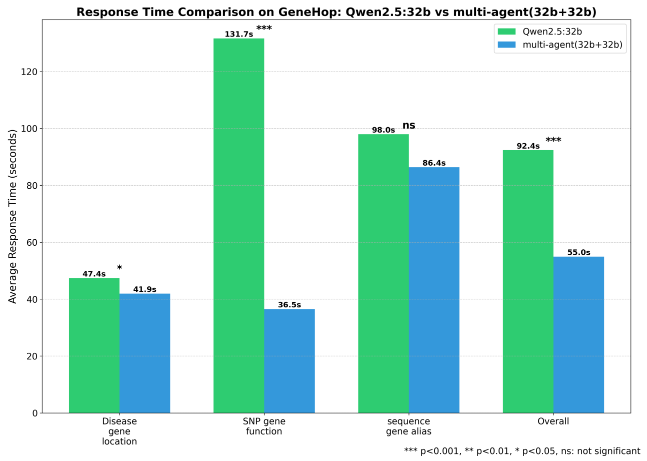

Figure 4: Task-wise latency comparison on GeneHop: Qwen2.5-32B vs. Multi-Agent (32B+32B).

Protein-coding genes, with improvements often exceeding 30–40% ($p < 0.001$). Gains are largely attributable to the modular execution strategy. Although the overall model size remains the same, each agent handles a narrower subtask with shorter prompts and limited context, which in turn reduces individual model response time and improves overall pipeline efficiency.

In GeneHop (Figure 4), the benefits are even more pronounced. The most dramatic example is SNP gene function, where the monolithic model takes over 130 seconds on average, while the multi-agent pipeline reduces it to just 36.5 seconds. This illustrates how task decomposition not only enhances reasoning accuracy but also leads to substantial speedups by avoiding overloading a single model with multi-step reasoning and large intermediate context.

Although a few tasks (e.g., Gene disease association, Sequence gene alias) show non-significant differences, the overall trend confirms that modular multi-agent architectures can achieve lower latency even when using the same underlying model capacity. This supports our claim that architectural design plays a critical role in the efficiency of LLM-based systems.

8.2 Other Advantages

Model Allocation: By decomposing the reasoning workflow into specialized sub-tasks, OpenBioLLM allows each component to leverage the strengths of different models. High-capacity models like 32B are assigned to handle complex decision-making and answer generation, while lightweight 14B models take care of simpler structured queries, such as parameter extraction and API formatting. This separation not only improves overall efficiency, but also opens the door for further specialization. For example, each agent can be replaced with a model fine-tuned specifically for its task. In this way, the system maximizes the utility of available models by aligning their capabilities with task complexity.

Scalability: Unlike monolithic designs that rely on hardcoded, fixed workflows, our architecture supports modular expansion. New tool agents—such as specialized APIs or knowledge bases—can be integrated seamlessly by extending the router's decision space. This makes OpenBioLLM adaptable to new domains or tasks without retraining the entire system.

Traceability: Another major advantage of the multi-agent setup is the ability to trace and debug the decision process. Each agent interaction is explicitly recorded, and using the LangSmith platform, we can inspect intermediate steps, tool invocations, and LLM responses. This visibility facilitates both evaluation and troubleshooting, and significantly improves the reliability of deployment.

9 CONCLUSION

This study explored the feasibility of open-source large language models (LLMs) for genomic QA. Building on GeneGPT, we developed **OpenBioLLM**, a modular multi-agent system that integrates tool augmentation, structured reasoning, and transparent API interaction.

We first reproduced the GeneGPT pipeline using Qwen2.5 and Llama3.1, identifying limitations in prompt design, tool invocation, and output formatting. Through prompt optimization and structured tool integration, we improved performance on both GeneTuring and GeneHop benchmarks, with Qwen2.5-72B achieving average scores of 0.86 and 0.84 respectively—surpassing the original Codex-based system.

To reduce token overhead and enhance interpretability, we introduced a modular architecture assigning different models to roles such as pipeline controller, tool agents, and evaluator. This 32B+14B configuration outperformed larger monolithic models, demonstrating the benefits of task decomposition and capacity specialization.

Error analysis showed most failures were due to insufficient evidence or parameter errors in complex tasks, rather than model limitations. Latency and routing evaluations further confirmed the architecture's efficiency, with faster responses and fewer tool calls per task.

Overall, OpenBioLLM presents a scalable and interpretable approach for applying open-source LLMs to genomics. Deployed within OpenWebUI, it enables real-time interactive exploration. We hope this work bridges the gap between general-purpose LLMs and the specialized needs of biomedical research.

REFERENCES

[1] Anthropic. 2024. Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet.

[2] Michael Chiang and Jeffrey Morgan. 2024. Ollama. https://github.com/ollama/ollama

[3] Google. 2024. Our next-generation model: Gemini 1.5. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024

[4] Jie He, Nan Hu, Wanqiu Long, Jiaoyan Chen, and Jeff Z. Pan. 2025. MINTQA: A Multi-Hop Question Answering Benchmark for Evaluating LLMs on New and Tail Knowledge. arXiv:2412.17032 [cs.CL] https://arxiv.org/abs/2412.17032

[5] Wenpin Hou, Xinyi Shang, and Zhicheng Ji. 2025. Benchmarking large language models for genomic knowledge with GeneTuring. *bioRxiv* (Jan 2025). https://doi.org/10.1101/2023.03.11.532238

[6] 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 (02 2024), btae075. https://doi.org/10.1093/bioinformatics/btae075 arXiv:https://academic.oup.com/bioinformatics/article-pdf/40/2/btae075/56803364/btae075.pdf

[7] Yevhen Kostiuk, Oxana Vitman, Łukasz Gagała, and Artur Kiulian. 2025. The Veln(ia)s is in the Details: Evaluating LLM Judgment on Latvian and Lithuanian Short Answer Matching. arXiv:2501.09164 [cs.CL] https://arxiv.org/abs/2501.09164

[8] Dan Li, Leihong Wu, Ying-Chi Lin, Ho-Yin Huang, Ebony Cotton, Qi Liu, Ru Chen, Ruihao Huang, Yifan Zhang, and Joshua Xu. 2024. Enhancing pharmacogenomic data accessibility and drug safety with large language models: a case study with Llama3.1. *Exp Biol Med (Maywood)* 249 (2024), 10393. https://doi.org/10.3389/ebm.2024.10393

[9] Jerry Liu. 2024. LlamaIndex. https://www.llamaindex.ai/

[10] Yizhen Luo, Jiahuan Zhang, Siqi Fan, Kai Yang, Yushuai Wu, Mu Qiao, and Zaiqing Nie. 2023. BioMedGPT: Open Multimodal Generative Pre-trained Transformer for BioMedicine. arXiv:2308.09442 [cs.CE] https://arxiv.org/abs/2308.09442

[11] Mistral. 2024. Cheaper, better, faster, stronger: Continuing to push the frontier of ai and making it accessible to all. https://mistral.ai/news/mixtral-8x22b

[12] OpenAI. 2024. Hello GPT-4o, 2024a. https://openai.com/index/hello-gpt-4o/

[13] Jebish Purbey, Siddhant Gupta, Nikhil Manali, Siddartha Pullakhandam, Drishti Sharma, Ashay Srivastava, and Ram Mohan Rao Kadiyala. 2024. SeQwen at the Financial Misinformation Detection Challenge Task: Sequential Learning for Claim Verification and Explanation Generation in Financial Domains. arXiv:2412.00549 [cs.CL] https://arxiv.org/abs/2412.00549

[14] Qwen. 2024. Qwen2.5: A party of foundation models. https://qwenlm.github.io/blog/qwen2.5

[15] Eric W Sayers, Richa Agarwala, Evan E Bolton, J Rodney Brister, Kathi Canese, Karen Clark, Ryan Connor, Nicolas Fiorini, Kathryn Funk, Timothy Hefferon, et al. 2019. Database resources of the National Center for Biotechnology Information. 47 (2019), D23–D28. Issue D1. https://doi.org/10.1093/nar/gky1069

[16] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2022. Large Language Models Encode Clinical Knowledge. arXiv:2212.13138 [cs] http://arxiv.org/abs/2212.13138

[17] Kaiming Tao, Jinru Zhou, Zachary A. Osman, Vineet Ahluwalia, Chiara Sabatti, and Robert W. Shafer. 2024. Fine-tuned large language models for answering questions about full-text biomedical research studies. https://doi.org/10.1101/2024.10.28.24316263 Pages: 2024.10.28.24316263.

[18] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971 [cs.CL] https://arxiv.org/abs/2302.13971

[19] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. BloombergGPT: A Large Language Model for Finance. arXiv:2303.17564 [cs.LG] https://arxiv.org/abs/2303.17564

[20] Hang Yang, Hao Chen, Hui Guo, Yineng Chen, Ching-Sheng Lin, Shu Hu, Jinrong Hu, Xi Wu, and Xin Wang. 2025. LLM-MedQA: Enhancing Medical Question Answering through Case Studies in Large Language Models. arXiv:2501.05464 [cs.CL] https://arxiv.org/abs/2501.05464

[21] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*.

[22] Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, et al. 2025. ToolHop: A Query-Driven Benchmark for Evaluating Large Language Models in Multi-Hop Tool Use. arXiv:2501.02506 [cs.CL] https://arxiv.org/abs/2501.02506

[23] Linan Yue, Qi Liu, Lili Zhao, Li Wang, Weibo Gao, and Yanqing An. 2024. Event Grounded Criminal Court View Generation with Cooperative (Large) Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. 2221–2230. https://doi.org/10.1145/3626772.3657698

A PROMPT DESIGN IN OPENBIOLLM

A.1 ReAct-style Prompting

Question: List chromosome locations of the genes related to Hemolytic anemia due to phosphofructokinase deficiency. Let us decompose the question to sub-questions and solve them step by step:
Thinking 1: What is the OMIM id of Glycine N-methyltransferase deficiency? [{url_8}] → [{response8}]
Action 1: Extract the OMIM id from the response: [606664,606628,601240,160993].
Thinking 2: What are genes related to OMIM id 606664,606628,601240,160993? [{url_9}] → [{response9}]
Action 2: Extract gene symbol from the response: [GNMT].
Thinking 3: What is the gene id of GNMT? [{url_10}] → [{response10}]
Action 3: Extract gene id from the response: [27232,14711,25134,41561,403338,697722].
Thinking 4: What is the chromosome location of GNMT? [{url_11}] → [{response11}]
Action 4: Extract maplocation as the answer.
Answer: 6p21.1

A.2 Router Module Prompt Details

You are a strict router that uses JSON to make routing decisions. You should analyze the conversation history above to make an informed decision about which agent to use.
router options:
- eutils_agent: query the database to get the detail information about gene, protein, disease.
- blast_agent: check the DNA sequence alignment and comparison.
- search_agent: search the web when you find there are a lot of eutils or blast call in conversation history but still cannot answer the question.
- irrelevant_questions: the question is not related to bioinformatics.

You should consider the following:
- What is the question?
- What information we have gathered so far?
- Which tool would be most appropriate for the next step?
- The evaluator’s opinion.

USER QUESTION:
{original_question}
CONVERSATION HISTORY:
{history_text if history_text else "No previous interaction history."}
PREVIOUS EVALUATOR’S OPINION:
{eval_reason else "No previous evaluator’s opinion."}
You MUST output your decision in the following JSON format:
"agent": "eutils_agent" or "blast$_a gent”or”search\_agent”or”irrelevant\_questions”,”reason”:$
$”Briefexplanationofwhythisagentwaschosenforthenextstep”$
}
Do not include any other text or formatting. ONLY return the JSON object.

A.3 Evaluator Prompt Details

A.4 Generator Prompt Details

A.5 E-Utils Prompt Details

A.6 Blast Prompt Details

A.7 Search Agent Output