On the Reproducibility of Superhuman Performance in Agentic RAG Systems for Scientific Question Answering

Phong-Anh Nguyen Trinh[1]

[1]Department of Physics, University of Cambridge

June 27, 2025

1 Introduction

Large Language Models (LLMs) have the potential to help scientists with retrieving, synthesizing, and summarizing the scientific literature [?]. However, issues such as hallucinations [?] [?], and underdeveloped retrieval and reasoning benchmarks, hamper the direct use of LLMs in scientific research .

The field is rapidly developing, with new models such as Google's Gemini 2.5 and OpenAI's o3 being able to 'reason', and excel at coding, maths, and langaage benchmarks. This also highlights the developments of new cutting-edge benchmarks for scientific performance in areas such as scientific discovery [?], analysis [?], programming [?], [?], and mathematics [?].

Alongside the development of the fundamental models themselves, techniques such as Retreival Augemented Generation (RAG) and the use of Multi-Agent Systems (MAS) allowed better use of LLMs in scientific research.

1.1 Retrieval Augmented Generation

Retrieval Augmented Generation (RAG) is a sophisticated technique designed to enhance the capabilities of Large Language Models (LLMs) by grounding their responses in external knowledge sources. This process mitigates common LLM issues like hallucinations and outdated information by dynamically providing relevant, fact-based context. The RAG process can be broken down into several key stages [?] [?]:

1. **Indexing**: A corpus of documents (e.g., scientific papers, reports, web pages) is processed into a searchable format. This involves loading the documents, splitting them into smaller, manageable chunks of text, and converting these chunks into numerical representations called vector embeddings using an embedding model. These embeddings are then stored in a specialized vector database, which allows for efficient similarity searching.
2. **Retrieval**: When a user submits a query, it is also converted into a vector embedding using the same model. The vector database is then searched to find the document chunks with embed-

dings that are most similar to the query embedding. This similarity search efficiently identifies the most relevant information in the knowledge base.

3. **Augmentation and Generation**: The retrieved document chunks are then combined with the original user prompt and fed as context to an LLM. The model uses this augmented prompt to generate a response that is synthesized directly from the provided information, ensuring the answer is relevant, accurate, and grounded in the source material.

This architecture is powerful because it separates the knowledge base from the generative model, allowing the knowledge to be updated without needing to retrain the LLM.

INCLUDE A PLOT AND EXPLAIN HOW RAG WORKS

1.2 Multi-Agent Systems

Multi-Agent Systems (MAS) represent a shift from using a single LLM to orchestrating multiple, specialized LLM-powered agents that collaborate to solve complex problems. This approach is analogous to a team of human experts, where each member has a specific role and set of tools. Instead of being "trained" in the traditional sense, these agents are *instructed* through carefully crafted prompts that define their roles, goals, and constraints. Their power comes from a well-defined system of interaction and tool use [**?**].

The development of a MAS typically involves three key components:

1. **Role Specialization**: Each agent is given a specific persona or role (e.g., "Planner," "Code Critic," "Financial Analyst") through its system prompt. This focuses the agent's behavior on a specific part of the problem.
2. **Tool Use**: Agents are equipped with specific tools, such as web search APIs, code interpreters, or private knowledge bases, allowing them to interact with the outside world and perform actions beyond text generation.
3. **Orchestration/Planning**: A framework is used to manage how the agents collaborate. This can be a hierarchical structure where a "manager" agent delegates tasks, a sequential process where tasks are passed down a pipeline, or a conversational "group chat" where agents can interact more dynamically.

Several powerful frameworks have emerged to facilitate the creation of these systems. **Microsoft's AutoGen** [**?**] excels at creating flexible, conversation-based agents that can dynamically interact. **LangGraph**, [**?**] an extension of the popular LangChain library, provides more explicit control by allowing developers to define workflows as stateful graphs, which is ideal for complex and cyclical processes. **CrewAI** offers a more intuitive, high-level abstraction, focusing on creating a "crew" of agents with predefined roles to tackle tasks in a structured, sequential manner. These tools mark a major leap forward toward automated scientific discovery and complex problem-solving.

1.3 PaperQA2

To make scientific discoveries, one must be able to synthesise scientific knowledge. Skarlinski et. al. believe this can be broken down into three vital tasks: scientific question answering, summarising, and detecting contradictions in the literature. In their paper, it is shown that their developed tool, PaperQA2, outperforms the human benchmark in all three areas [**?**]. The focus of this report is to understand and reproduce the question-answering result.

1.3.1 PaperQA2 Architecture

PaperQA2 operates as an agentic Retrieval-Augmented Generation (RAG) system built upon the `paperqa` Python package. Its architecture enables a flexible workflow where an agent can operate on a pre-indexed local corpus or dynamically search for and ingest new documents at query time. The typical agentic query process involves the following stages:

1. Paper Search and Dynamic Ingestion: The agent's workflow begins by invoking a search tool, such as one that interfaces with the `Semantic Scholar` API. The agent first uses an LLM to distill the user's query into a set of precise keywords. These keywords are used to query the external academic database, which returns a list of candidate papers. The agent then retrieves the relevant documents (e.g., as PDFs) and parses them to extract their content. While the framework includes a built-in `PyMuPDF` parser for text extraction, it can also be integrated with more advanced external tools like `Grobid`, which can parse the full document structure (e.g., title, sections, references). The extracted text is then segmented into configurable, overlapping chunks. These chunks are immediately converted into vector embeddings using a configurable model and indexed in both a keyword store (`tantivy`) and a vector store (`Numpy` or `Qdrant`). This on-the-fly ingestion process makes the newly found papers available for the subsequent stages of the current query.

2. Evidence Gathering and Re-ranking: With a set of newly ingested chunks available, the system proceeds to a multi-step evidence-gathering phase to refine the context. This involves:

- **Vector and Hybrid Search:** A vector cosine similarity search is performed on the candidate chunks. The framework supports **hybrid search**, a technique that combines two different kinds of vectors for ranking. It uses dense vector embeddings, which capture the semantic or conceptual meaning of the text, along with sparse keyword-based vectors (e.g., TF-IDF or SPLADE-style models?), which excel at exact term matching. By merging scores from both, hybrid search retrieves chunks that are both thematically related and contain precise keywords from the query.
- **Result Diversification:** To prevent the retrieval of highly redundant information from a single source, the system can apply **Maximal Marginal Relevance (MMR)**. Instead of just optimizing for similarity to the query, MMR iteratively selects chunks by balancing their relevance with their novelty compared to chunks already selected. This ensures the resulting evidence set is both on-topic and informationally diverse.
- **LLM-based Re-ranking and Contextual Summarization (RCS):** This is arguably the most critical and computationally intensive step, invoked by a `GatherEvidence` tool. The top k chunks from the previous steps (a number configurable via `answer.evidence_k`) are not used directly. Instead, each individual chunk is passed to an LLM with a specific prompt instructing it to summarize the chunk's content strictly in relation to the user's original query and to assign a relevance score. This use of an LLM as a "re-ranker" is highly effective at filtering out passages that may be semantically similar but not actually useful for answering the question, while simultaneously distilling the most important information.

3. Answer Generation: In the final stage, a `GenerateAnswer` tool compiles the highest-scoring summaries from the RCS step into a single context. The number of evidence pieces included is configurable via parameters like `answer.answer_max_sources`. This context, along with the original question, is formatted into a final prompt using a customizable template (`prompt.qa`). This allows for fine-grained control over how the LLM is instructed to synthesize the information. The model then generates a comprehensive answer that is directly grounded in the selected sources. The system formats this answer with inline citations that trace back to the original documents, providing a verifiable response.

4. Citation Traversal: As an optional step, the agent can employ a **Citation Traversal** tool to expand the scope of its search beyond the initial document corpus. After identifying a highly relevant paper, this tool can be used to analyze its bibliography or query external databases (like Semantic Scholar) to find papers it cites (backward traversal) or papers that cite it (forward traversal). This is a technique for discovering foundational research or, conversely, the latest developments building upon a known paper. These newly discovered documents can then be ingested and indexed as it runs, allowing the agent to dynamically expand its knowledge base to answer a query more comprehensively.

1.3.2 Key Result

The paper claims that PaperQA's performance beats the human benchmark, specifically for the precision of questions answered, achieving a precision of 85.2% compared to the human precision of 73.8&. The benchmark metrics are accuracy and precision.

$$Accuracy = \frac{CorrectQuestions}{AllQuestions} \tag{1}$$

$$Precision = \frac{CorrectQuestions}{AnsweredQuestions} \tag{2}$$

The process of evaluating the performance of PaperQA2 was to ask questions, where the answers were found in newly published papers and could not be inferred from either the title, abstract, or conclusion. The questions would also require some 'reasoning' and the answer could not be a direct quote from the paper. The questions were taken from recently published biology papers, post September 2021 (the GPT training cutoff). The assumed reason for this is to prevent the LLM agents from using their 'own knowledge', instead of reading and understanding the papers given to it.

Human evaluators were then tasked with answering the same questions. However, the human evaluators were not directly given the correct paper (or the correct paper within a group of papers - which is essentially how PaperQA performs evaluation). Instead, the evaluators were access to the internet and journal collections, but were told to not use generative AI tools such as ChatGPT.

The aim of this project is to recreate this result, and confirm whether language agents can outperform human performance in question answering.

2 Methods

2.1 Data Analysis

The first step was to collect and assemble the dataset for the LitQA benchmark. The questions, answers, and distractors, as well as the paper DOIs, were provided on FutureHouse's HuggingFace repository. Initially, two main issues were faced: only the training dataset was available, and the papers themselves were not provided. The test set was also difficult to access but was eventually located. Subsequently, using the collected DOIs, all individual paper PDFs were downloaded and assembled into the final datasets.

The next step was to analyse the papers themselves. During the collection process, it was noted that some of the paper publication dates did not meet the requirements stated by the paper's authors. This was a critical issue, as it could affect the validity of the results. Consequently, the invalid papers were identified; it was found that 15 papers preceded the specified cutoff date. Figure **??** shows the publication years of the papers. These papers were subsequently excluded from the dataset, and a 'valid' subset of PDFs was created for the experiments.

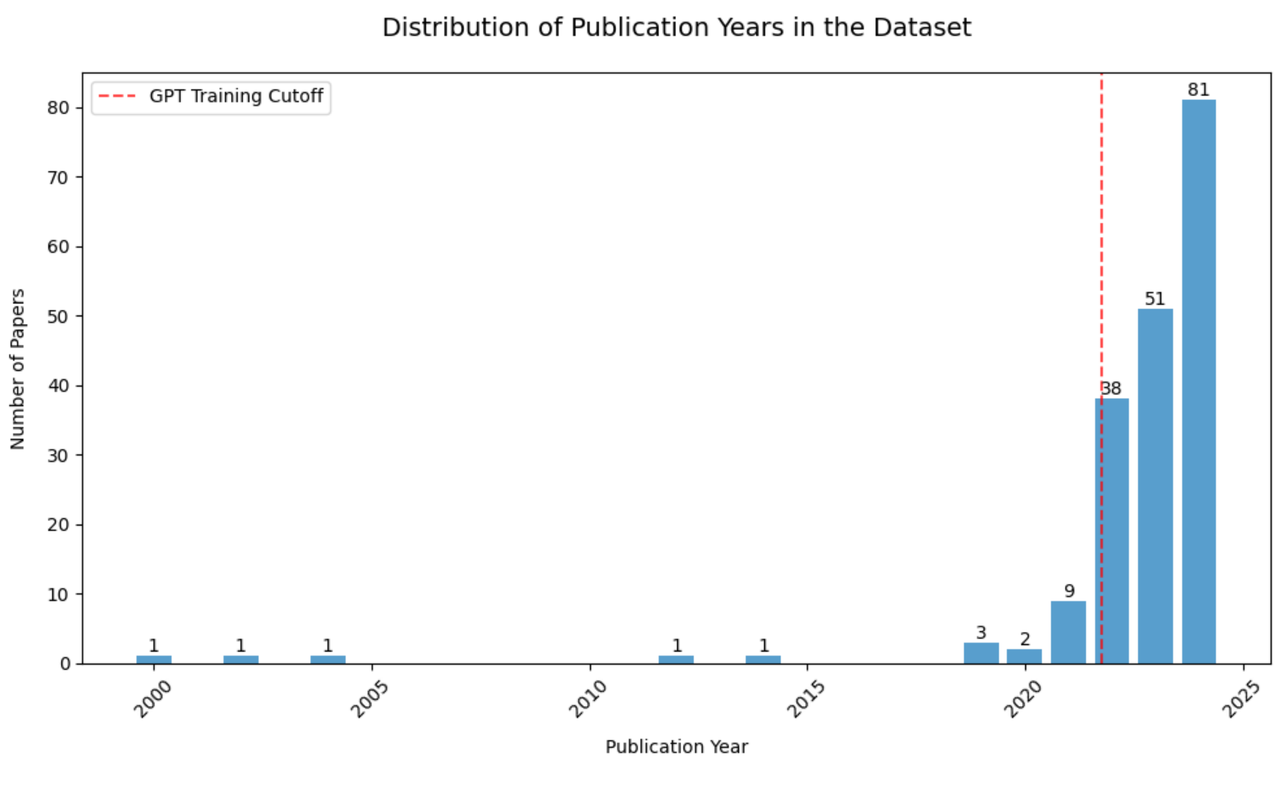


Figure 1: Distribution of the publication years of all of the papers in the LitQA dataset. The red line represents the training cutoff date for GPT.

The data from HuggingFace, which contained the question, correct answer, and distractors, was loaded into a `pandas` DataFrame. For each question, the correct answer and distractors were shuffled and then assigned a letter to formulate a multiple-choice question.

2.2 Result Reproduction

After the dataset was finalized, the next step was to reproduce the original study's results using PaperQA. The package provides an intuitive interface where the user calls the `ask` function (a synchronous wrapper for the asynchronous function `agent_query`). This function uses the user's prompt to find relevant documents to answer the query. Each question was fed to the `agent_query` function, and the resulting answer was recorded.

The standard output of PaperQA is a verbose summary that includes the answer, the reasoning behind the selection, and citations from the source papers. While this is a highly useful feature for researchers, its verbosity posed a challenge for the automated evaluation required by this project, which necessitated a single-letter answer for direct comparison against the ground truth. Initially, prompt engineering was employed to compel PaperQA to return only a single letter.

```
The following is a multiple choice question about biology.
Please answer by responding with the letter of the correct answer.

Think step by step.

{question}
```

Where the question would contain the question and choices:

```
Question: Approximately what percentage of topologically associated
domains in the GM12878 blood cell line does DiffDomain classify as
reorganized in the K562 cell line?
    A) 11%
    B) 41%
    C) 21%
    D) 51%
    E) 31%
    NA) Insufficient information to answer the question.
```

Although this method produced a single-letter output, the agent invariably returned a short summary explaining its reasoning. This led to inconsistent outputs, where the position of the single-letter answer varied within the text, making it difficult to create a reliable evaluation pipeline.

The solution was to leverage structured outputs, which allow users to customize the exact output format of an LLM. However, PaperQA internally uses `LiteLLM` to invoke LLMs for tasks such as paper retrieval, evidence gathering, and answer generation. As structured output formats vary between different LLM providers and can typically only be used when calling models directly, there is no native entry point to enforce structured outputs within the PaperQA framework.

2.3 Custom Multi-Agent Wrapper for PaperQA

The solution to the structured output problem was to employ a multi-agent system, mirroring the architectural philosophy of PaperQA itself. Enforcing a JSON-style structured output was made possible by using a secondary agent system. Using a foundational 'ConversableAgent', a simple linear agentic system was developed to standardize the format of both the input to and the output from PaperQA. This ensured a consistent output format for evaluation. A diagram of this agent structure is required here. The requirement for a standardized input format was driven by the specific needs of the evaluation methodology.

2.4 Evaluation and Inspect AI

Inspect AI is a framework developed by the UK's AI Security Institute for evaluating the performance of LLMs. Its `eval` tool facilitates rapid performance evaluation by running LLM calls asynchronously. An `eval` task consists of three components: a dataset, a solver, and a scorer. The dataset is composed of `Sample` objects, each corresponding to an individual task (in this case, a multiple-choice question) and containing fields for the input, choices, and target answer. The solver represents the LLM system being evaluated, and the scorer defines the evaluation metric. A key benefit of Inspect AI is its provision of an intuitive interface for monitoring evaluation progress.

A separate custom library, `inspect_evals`, exists for evaluating the performance of various LLMs on benchmarks such as LitQA [**?**]. However, it is only compatible with single LLMs, not multi-agent systems.

The `inspect_ai` framework does offer support for multi-agent systems via its `bridge` function. However, the package is still in early development and, in its native form, does not support end-to-end evaluation of multi-agent systems on multiple-choice questions. The `bridge` function effectively allows a multi-agent system to act as the solver, but its communication format only supports the passing of agent messages, not structured data like the question, choices, or target answer. This limitation prevents the scorer from accessing the necessary information to determine if an answer is correct.

To overcome this, a custom package for multi-agent MCQ evaluation was developed using `inspect_ai`. This package integrates with Inspect AI and allows any multi-agent system to be evaluated, provided it accepts a question and returns a text reply. It leverages a custom dataset builder that reads the `pandas` DataFrame and creates custom `Sample` objects with a structured JSON format. These samples are passed to a custom bridge agent that encapsulates the multi-agent system. The resulting answer is then evaluated by a custom scorer. Answers are classified as CORRECT, INCORRECT, or NA (for cases where the agent fails or cannot find an answer). This implementation was based on the logic in the `inspect_evals` package. The custom scorer parses the structured output from the agent system and converts the data into an `inspect_ai.Score` object, allowing the `eval` tool to calculate and aggregate performance metrics across the entire task. Below is an example prompt would be given into the `bridge` agent, which would partition the information and ensure the correcrt information is passed to the appropriate part of Inspect AI

```
Question: Approximately what percentage of topologically associated
domains in the GM12878 blood cell line does DiffDomain classify as
reorganized in the K562 cell line?
    A) 11%
    B) 41%
    C) 21%
    D) 51%
    E) 31%
    NA) Insufficient information to answer the question.
    Target) E
```

2.5 Hyperparameters

The main hyperparameter varied in this project was the LLM itself. In the original paper, GPT-4-Turbo was the default model. However, given the rapid pace of model development, several newer models were tested to assess performance improvements. The models evaluated were OpenAI's GPT-4o-Mini, GPT-4.1, and GPT-4-Turbo. Due to budget constraints, the default model used in our experiments was GPT-4o-Mini, which offers performance comparable to or better than GPT-4-Turbo for a lower per-token cost.

Another hyperparameter that was changed was the text embedding model. The default model used by PaperQA is OpenAI's `text-embedding-3-small`, which was no longer state-of-the-art at the time of the experiments. Instead, the highest-performing model available via LiteLLM, Google's `text-embedding-004`, was used. Embeddings are crucial for NLP tasks, as a more accurate semantic representation of text is hypothesized to improve an LLM's comprehension and subsequent decision-making.

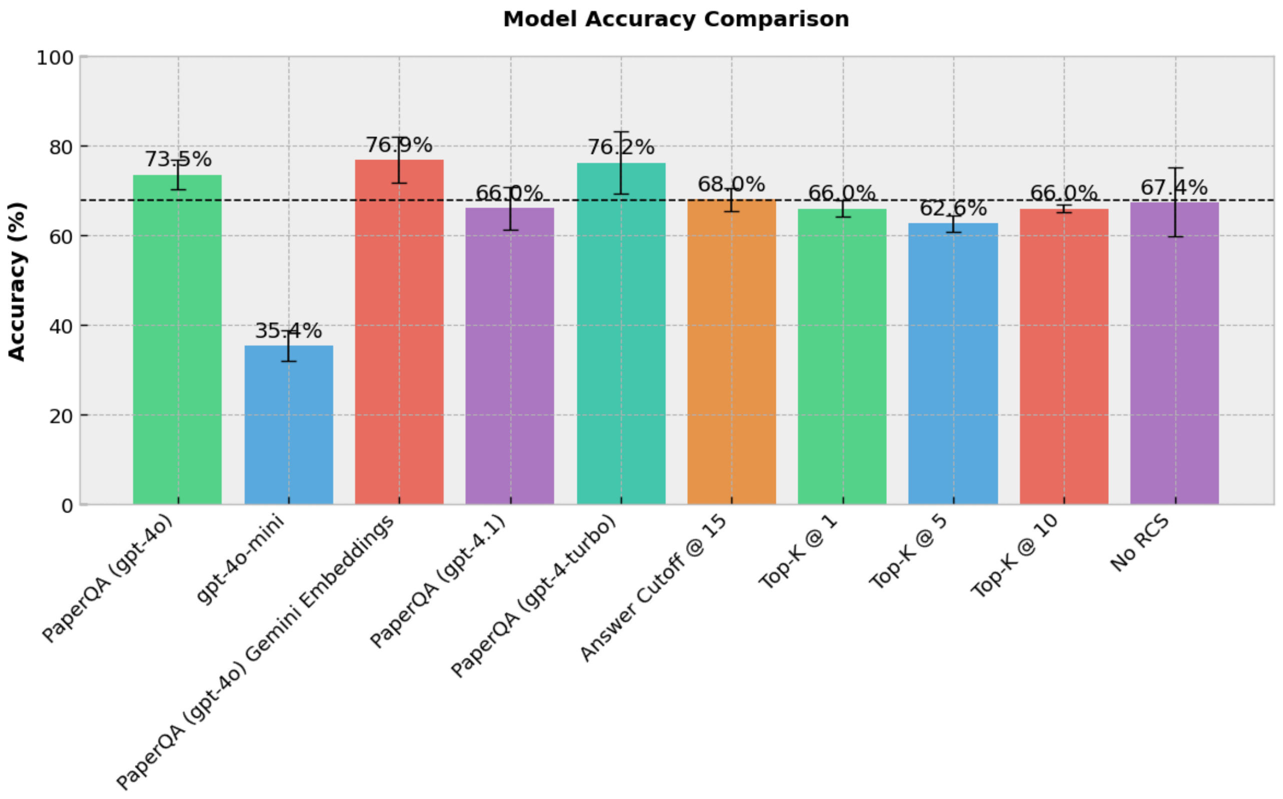
Within PaperQA, there are also two key parameters that the original authors varied. First is the Answer Cutoff (`max_sources`), which limits the maximum number of retrieved sources used to generate a response. This filtering occurs after the RCS step. The other hyperparameter is `evidence_k` ('Consider Sources'), which controls the initial document retrieval step. To use an analogy: a researcher might retrieve the 30 most promising articles from a library (`evidence_k` = 30). From these, they might skim the articles and select the best 5 (`max_sources` = 5) to read thoroughly and inform their final answer. In the original paper, the default values for `max_sources` and `evidence_k` were 5 and 30, respectively. If `evidence_k` is set lower than `max_sources`, then `max_sources` is automatically reduced to match. The authors also tested an `max_sources` of 15 but found its performance to be worse; therefore, 5 was used as the default in our experiments.

PaperQA differentiates itself from other RAG systems by employing RCS. To test the criticality of this step, an experiment was also performed where RCS was disabled.

Other hyperparameters, such as temperature, were kept consistent throughout the experiments to make the results as reproducible as possible.

3 Results

Figure **??** displays the results from the experiments conducted in this project, while Figure **??** shows the results from the original paper for comparison. This section examines and contrasts these two sets of results. All experiments were run three times to assess the performance variation of the different ablations.



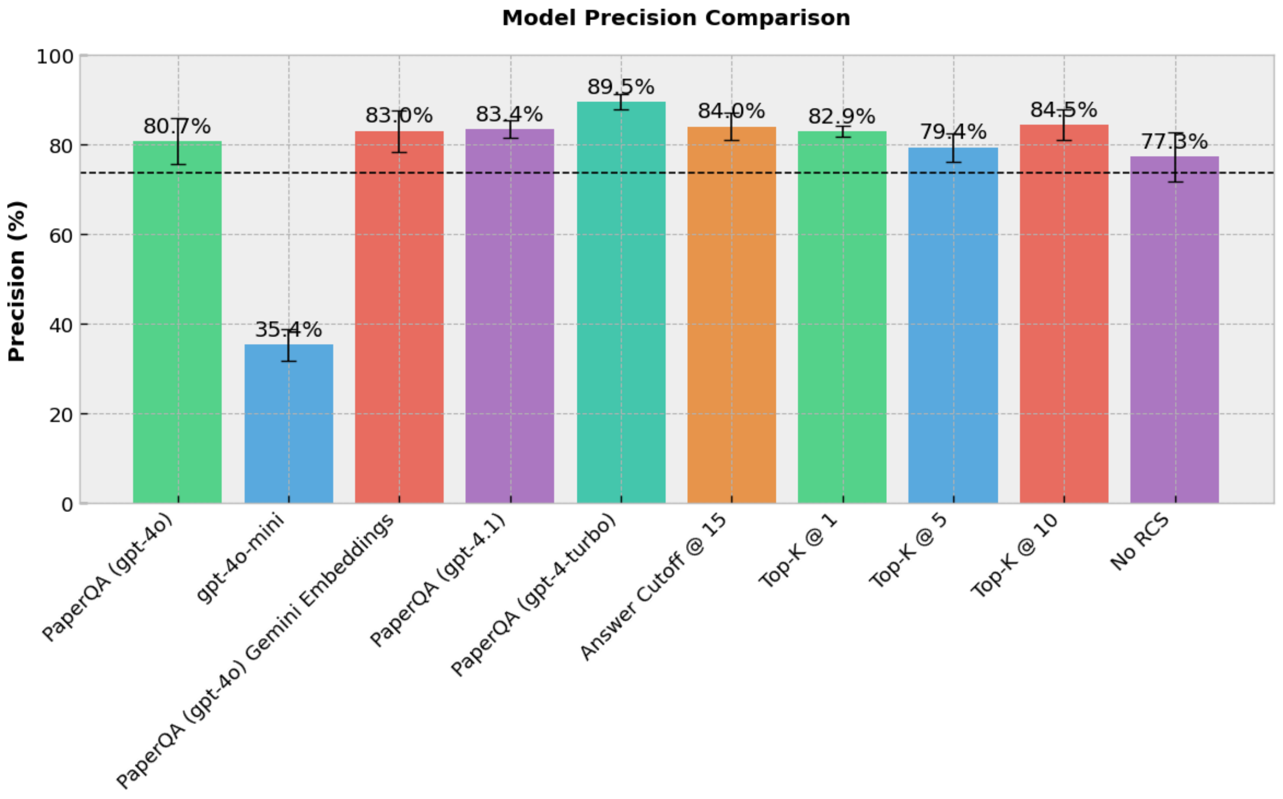


Figure 2: This project's results against the LitQA test set. Top: Average accuracy of various setups of PaperQA. Bottom: Average precision of the various setups of PaperQA. The dashed line in each plot represents the human benchmark for both accuracy and precision.

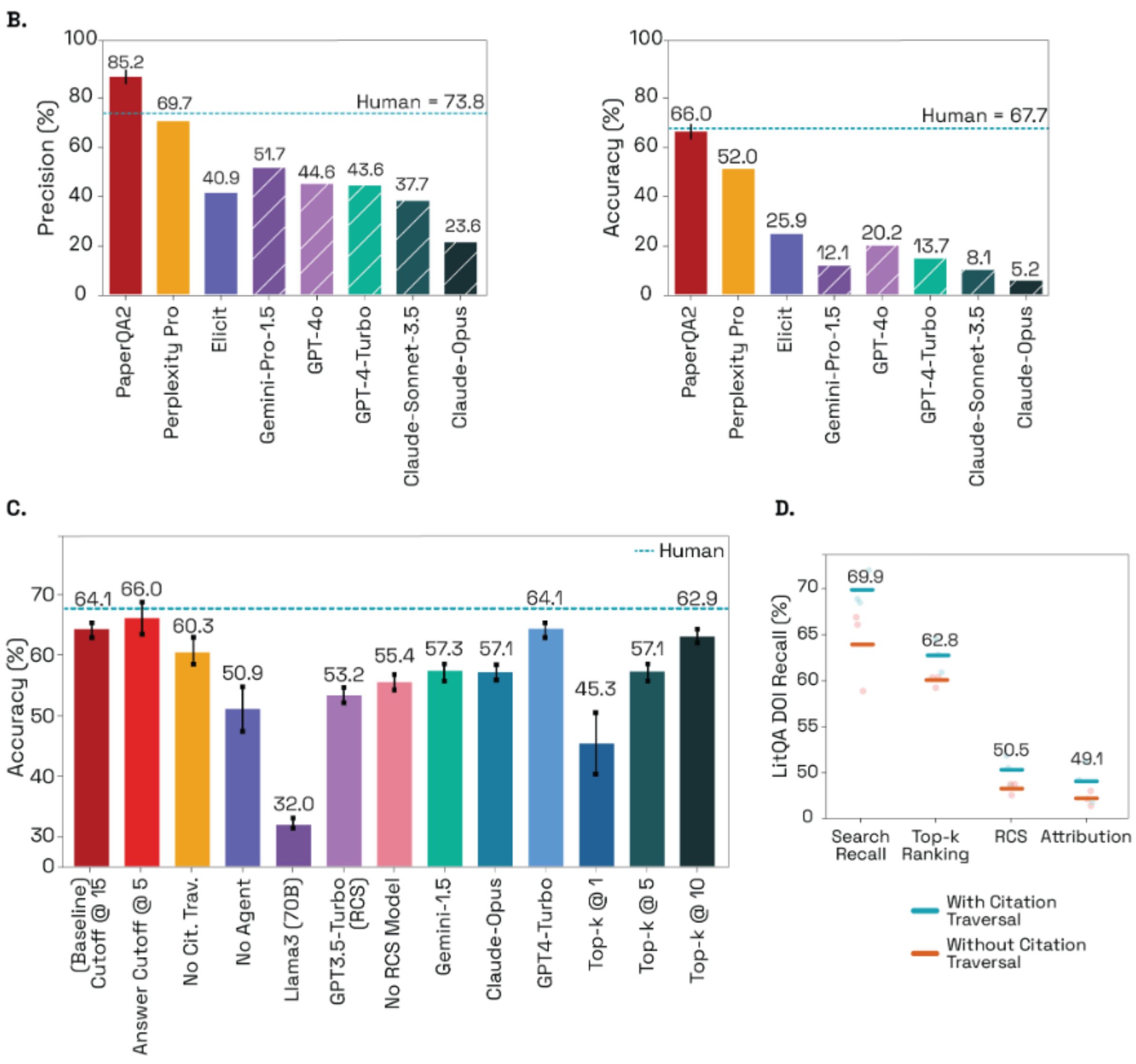


Figure 3: Original Paper Results. B: Precision and Accuracy of PaperQA and LLMs on LitQA. C: Accuracy of various ablations of PaperQA. D: Aggregated DOI Recall at each stage (This result is not relevant to this report.)

3.1 RAG vs. Standalone LLM

Consistent with the original study, the results from this project demonstrated that PaperQA outperforms standalone LLMs in question answering. A key difference in behavior was observed: in all standalone evaluations, the LLM attempted to answer every question. Consequently, its accuracy and precision scores were identical, leading to a high rate of hallucination as the model would guess rather than concede an inability to answer. The standalone LLM used in this experiment was `gpt-4o-mini`, whereas the original paper utilized `gpt-4-turbo`. The performance of `gpt-4o-mini` was slightly worse in terms of overall accuracy.

3.2 PaperQA Ablations

The following experiments tested the effect of changing key hyperparameters of PaperQA, from the LLMs used for the agent and RAG components to the top-k retrieval and answer cutoff settings.

The recreated baseline for PaperQA utilized `gpt-4o-mini` as the agent and RAG model, with a top-k of 30 and an answer cutoff of 5. The text embedding model was the PaperQA default, `text-embedding-3-small`. In contrast, the original baseline used `gpt-4-turbo` for the agent and RAG model, with a top-k of 30 and an answer cutoff of 15. An answer cutoff of 5 was chosen for our baseline, as this was the configuration that achieved 'superhuman' performance in the original study. The recreated baseline appeared to exceed the original, achieving superhuman performance in both accuracy and precision.

To more closely replicate the original result, an experiment was run using `gpt-4-turbo` for both the agent and RAG model. The performance was expected to be similar to the original result; however, our experiment achieved superhuman performance in both accuracy and precision. An inspection of the results yielded no evidence of scoring errors that would explain this anomaly.

Unexpectedly, using PaperQA with `gpt-4.1` for both the agent and RAG model resulted in the accuracy dropping to sub-human levels compared to both the original result and our GPT-4-Turbo experiment, with the precision being similar but slightly lower. This was a surprising outcome, as GPT-4.1 is a newer model expected to outperform GPT-4-Turbo across standard benchmarks.

Across all model variations, the best-performing model in terms of accuracy was the GPT-4o-Mini configuration that used Google's state-of-the-art text embedding model, `text-embedding-004`. It outperformed the original baseline and consistently achieved superhuman performance in both accuracy and precision.

Increasing the answer cutoff had the same effect as in the original paper, decreasing accuracy. However, the effect of removing the RCS step was much less pronounced than in the original study; whereas the original paper reported an accuracy reduction of 12.8%, our experiment showed a reduction of only 6.1%.

The effect of changing the top-k retrieval setting had a much smaller impact on accuracy compared to the original paper. While the general trend in the original study was that decreasing the top-k value decreased accuracy, our experiments showed no clear correlation. In fact, lowering the top-k to below 10 still yielded near-superhuman accuracy, a level the original paper's configuration (with top-k=30) only narrowly achieved.

The RCS step was considered crucial for superhuman performance in the original paper, where its removal led to a decrease of around 10% in accuracy. In our recreated experiment, removing the RCS step led to only a 6% decrease in accuracy, with performance remaining very close to the human benchmark.

A key finding is that all experiments involving RAG significantly surpassed the human benchmark for precision, with all ablations achieving at least 77.3% precision, and the best-performing ablation achieving 89.5%, exceeding the human benchmark of 73.8%.

The accuracies of the ablations were more varied than the precisions, but this variation was less pronounced than in the original paper. All RAG configurations performed close to or better than the human benchmark, even with hyperparameter settings that were theoretically expected to hinder performance.

It should be noted that an error was discovered in the original accuracy calculation script. Conse-

quently, all accuracy metrics reported herein have been recalculated from the raw outputs to ensure correctness, obviating the need to re-run the experiments.

4 Discussion

4.1 Paper Methodology and Results

4.1.1 Human Benchmark

The human benchmark is the standard used to determine whether LLM agents achieve superhuman performance. The criteria for the questions proposed in the LitQA benchmark appear suitable; the requirements for papers to be recently published and from cutting-edge fields were mostly met, with the majority of publications originating from Nature and journals of equivalent standing. The stipulation that answers cannot be inferred from the abstract or title enforces that the questions are non-trivial and require a degree of reasoning. Furthermore, the use of distractors—incorrect options derived from related facts or statistics from the same paper—necessitates a deep understanding of the content rather than a simple keyword search. However, despite these criteria set by the authors, the dataset was curated by external contractors and, as demonstrated in the previous section, did not always adhere to these requirements, including several papers that predated the cutoff. Without the resources to manually verify every question, the integrity of the dataset, and consequently both the human and machine benchmarks, is subject to a degree of uncertainty.

Regarding the human evaluators' performance, their specific academic backgrounds were not provided, although it can be inferred that they were at least at the university level, as the authors stated they had access to journals "through their institutions." The evaluators were unrestricted in their methodology, with access to the internet and search engines, but not necessarily direct access to the source papers. This implies that a significant part of their task was information retrieval—finding the correct source to answer the question. This mirrors the process undertaken by the PaperQA agent, where performance is heavily dependent on successfully retrieving the correct evidence chunks. Thus, the human benchmark closely resembles the task faced by the agent.

To maximize performance, the evaluators were financially incentivized to answer as many questions correctly as possible within a one-week period. This approach establishes a high-performance benchmark, appropriate for testing claims of superhuman capabilities. While evaluators were instructed not to use generative AI tools, this was not enforceable. The financial incentive also introduces a risk of cheating, though this risk is likely low given the specialized nature of the questions. Having faith in humanity, the human benchmark was deemed a valid, albeit imperfect, standard for comparison.

4.1.2 Comparison with Original PaperQA Results

The results of this project revealed some major differences when compared to the original paper's experiments, stemming from both model and methodological variations.

A significant methodological difference was the environment used. The original authors employed a custom HTTPS environment with features such as MongoDB request caching, Redis object caching, global load-balancing, and cloud-based orchestration. They stated this infrastructure did not affect per-run results but enhanced scalability, measurability, and persistence. In contrast, this project was executed on a local machine with standard specifications, storage, and compute. While individual PaperQA runs were successful, the full evaluation process encountered significant issues. The majority of erroneous runs were caused by wall-time limit failures or API rate-limiting. The time limit was often exceeded because the local machine had to manage the `inspect_ai` evaluation framework concurrently with the agent calls, a compute-intensive process. Rate limit errors arose from running multiple agent calls in parallel from a single API key. These failures directly impacted the results; erroneous runs were marked as 'NA', which artificially lowered the accuracy score while potentially inflating the precision (as fewer questions were considered 'answered'). The authors' use of a custom infrastructure likely mitigated these issues, suggesting their reported results might not be representative of the performance an ordinary user would achieve on a local machine.

4.2 Project Results

4.2.1 Methodology Breakdown

The methods employed in this project aimed to identify and address potential sources of bias and uncertainty in the original study.

The first methodological difference was that our experiments were conducted exclusively on the test dataset, rather than the full LitQA benchmark. This decision was driven by several factors. Firstly, budgetary constraints prevented multiple runs on the full dataset. Secondly, the original authors had already identified that 147 questions in the benchmark had been compromised by Google's data aggregator. It is unknown if these questions have since been updated in the benchmark. More importantly, if this data was scraped by Google, it was likely also scraped by OpenAI and other providers to train their latest models, which defeats the purpose of testing the agents' ability to reason over novel scientific information. The use of a smaller evaluation dataset, however, introduces the inherent risk of higher variance in the results, as each question carries a greater weight in the final performance metrics.

Another potential source of error was the use of a secondary agent system for formatting. This is a common design pattern in multi-agent systems, where specialized agents handle tasks like I/O parsing. While this increases the robustness and consistency of the primary agent's output, it also introduces another component that could potentially fail or hallucinate, adding a layer of complexity to error analysis.

The use of `inspect_ai` was intended to follow standardized evaluation methods. However, as LLMs and agentic systems represent a new and rapidly evolving field, a standardized workflow for integrating and evaluating such complex, multi-component systems does not yet exist. Integrating PaperQA into our custom agent system and then into Inspect AI proved to be extremely difficult. Inspect AI itself is a new tool and lacks some of the functionality required for streamlined multi-agent evaluation. While it offers benefits like asynchronous execution, which theoretically should provide a speedup, it also reduces visibility into the underlying processes. Passing metadata through the evaluation pipeline was particularly challenging, necessitating the multi-agent wrapper and creating difficulties with cost tracking.

Regarding asynchronous execution, while theoretically beneficial, running PaperQA calls in parallel proved to be counterproductive in practice. It significantly increased the time for individual queries (from 1 minute to 15 minutes for just two or three parallel queries) and led to a higher rate of time-out and rate-limit errors. This is likely because parallel requests rapidly exhaust the API's rate limit quota, triggering exponential backoff and retry mechanisms in the client library. This "retry storm" results in long wait times that ultimately make sequential execution both faster and more reliable.

4.2.2 LLM Models and Embeddings

Varying the LLMs had a less significant effect on precision than on accuracy. It appears that once a RAG system retrieves the correct context, most modern models are proficient at extracting the correct answer. All tested RAG configurations achieved 'superhuman' precision, underscoring the value of grounding models in external evidence. This contrasts sharply with the baseline `gpt-4o-mini` model without RAG, which attempted every question, leading to identical (and low) accuracy and precision scores due to frequent hallucination. With RAG, the models became 'aware' of their knowledge boundaries, preferring to abstain from answering rather than guessing, a critical feature for scientific applications. While no model achieved 100% precision, the tendency to hallucinate was dramatically reduced compared to non-RAG performance.

The disparity between high variation in accuracy and low variation in precision suggests that the primary challenge for these systems is not reasoning over provided text, but successfully retrieving the correct text chunk in the first place. Once the relevant information was sourced, all models performed similarly well. This highlights that the most critical stage is information retrieval.

This conclusion is further supported by the strong performance of the configuration using Google's `text-embedding-004` model. Since all the papers in the LitQA corpus are from the field of biology, they are semantically similar, making it challenging for less advanced embedding models to distinguish between them. Better embeddings create more distinct clusters in the vector space for different nuanced topics. This reduces the "semantic noise" and helps the retriever pull more distinct and relevant chunks, leading to the highest accuracy observed.

A notable anomaly was the performance of `gpt-4.1`. Despite being a newer and more powerful model than `gpt-4-turbo`, it performed worse on this task, showing a greater hesitancy to answer and achieving lower accuracy. This suggests that general-purpose benchmark performance does not always translate to specialized, agentic RAG workflows and may indicate poorer performance in its function as either a retrieval or re-ranking agent. It is possible the model is over-optimized for conversational tasks at the expense of the kind of structured "tool-use" reasoning required by PaperQA. This behaviour remains unexplained, as the architectures of these proprietary models have not been released. Since the publication of the original paper, LLM research has advanced significantly, which is reflected in our results where nearly every RAG experiment achieved or exceeded human performance.

4.2.3 Hyperparameter Effects

The experimental results confirmed that system performance is most sensitive to hyperparameters controlling the retrieval process.

Increasing the answer cutoff (`max_sources`) from 5 to 15 led to a decrease in both accuracy and precision. This contradicts the naive assumption that more context is always better. This phenomenon is likely an example of the "lost in the middle" problem [?], where LLMs with large context windows struggle to attend to information buried in the middle of a long prompt. By expanding the number of sources, we increase the risk that the correct evidence, even if retrieved, is ignored by the model during the final synthesis step, leading it to answer incorrectly or hallucinate.

In the original paper, decreasing the top-k parameter led to a drastic reduction in accuracy. While our results reflected the same trend, the effect was far less pronounced. We found no clear correlation between the top-k value and accuracy, with newer models like `gpt-4o-mini` achieving near-superhuman performance even with a low top-k. This suggests that newer models are more effective in their initial retrieval step, identifying higher-quality, more relevant chunks from the outset. Consequently, the need for a large candidate pool for the subsequent RCS step is diminished.

This observation is further supported by the results from disabling the RCS step. In the original study, this was deemed critical for superhuman performance. In our experiments, its removal still yielded near-human performance, once again indicating that the improved baseline capabilities of newer models reduce the reliance on intensive re-ranking and filtering steps.

4.2.4 Issues

One very important issue noticed was the flucatuation in performance depending on LLM usage. Noticed quite late on in the project, after performing a sizable number of runs, it appeared that the performance of LLMs would change depending on when the tests would run even with temperature 0. Looking on forums [?], it appeared that this was a common problem, especially with OpenAI. Looking into the GPT-4 architecture could return a reason why this is happening. There is no GPT-4 techical paper that details its full archtitecture due to competition, but forums [?] , leaks and expert analysis [?] [?] [?] point to the fact that GPt-4 is a Mixture of Experts architecture. It is essentially an ensemble model of transformers and neural networks, each specialised for a 'expert' topic, rather than a general transformer, and the query is sent to a number of 'experts' to answer the question. The inference is then distributed to a number of computers and recollected for minimum computational load and maximum speed. However, it is this distributed factor that may cause the performance degradation at peak times. One reason is that even though we set temperature to 0, since the inference is distributed, it is not guaranteed that the same configuration is sent to each device (possibly due to compression etc.), and so floating point errors could accumulate and potentially lead to different outputs [?]. Looking at MOE, it could be possible that at peak times, when resources are contested heavily, models could be degraded to ensure everyone has access to the models (e.g. instead of 3 experts you are only allocated 2, also known as graceful degradation). Also, to balance the load, the experts could be given less time for the current user's prompt, so that it is freed for the next user's prompt, resulting in less time for processing. These are all potential ideas why this is happening and is not confirmed by providers such as OpenAI. This points to possibly using less popular providers for better performance.

There seemed to also be a disparity between the effect of hardware on the performance of the runs. Due to technical issues, some of the runs had to be run on two different laptops. One laptop had below average hardware specifications (no GPU, Intel 12th Gen 5 Series CPU), and the other had top of the line specifications (e.g. top-end AMD GPU and CPU). Despite running the exact same code, and the same version for the packages, there was a slight difference in performance. Predictably, the better laptop completed the evaluation of the test set quicker, but the difference in time was substantial (usually 35 minutes, instead of 50 minutes). Considering the code is primarily API calls, this was peculiar, as there was no difference in code or packages, and there seemed to be no part of the code that relied heavily on CPU or GPU performance.

The difference in specs did not affect single PaperQA runs, but it did affect the performance of evaluation runs of the full test dataset. The faster runs also often came with higher accuracy and roughly equal precision. This reinforces the conclusion that system performance is sensitive to the entire end-to-end execution environment, not just the model itself. These external factors could also provide an alternative explanation for the unexpectedly poor performance of the GPT-4.1 model; it is possible those specific runs were affected by higher API latency. There seemed to be no plausible explanation for this, and the only way to confirm this would be to perform an focused, individual study instead.

These implementation issues could have explained the poorer performance of GPT 4.1 compared to GPT-4-Turbo, which was expected to perform better. The results were not errorneous, as every single question had a reasonable answer, with an appropriate reason behind why the answer was given.

5 Conclusion

This report successfully reproduced and extended the evaluation of the PaperQA agent on the LitQA benchmark, revealing critical insights into the performance of modern agentic Retrieval-Augmented Generation (RAG) systems. The primary finding confirms the foundational premise of the original study: RAG systems significantly outperform standalone Large Language Models (LLMs), demonstrating superior precision by grounding responses in retrieved evidence and mitigating hallucination. Our experiments consistently achieved 'superhuman' performance that not only replicated but often exceeded the benchmarks set in the original paper, particularly when leveraging newer models like GPT-4o-mini and advanced text embeddings.

However, the investigation uncovered a more nuanced and complex reality. The results suggest a shift in the challenges of RAG systems; while the original paper emphasized the crucial role of the RCS step, our findings indicate that newer LLMs have diminished its importance. The primary performance bottleneck appears to have shifted from evidence re-ranking to the initial retrieval phase, where the quality of text embeddings and the inherent reasoning capabilities of the agent LLM are vital.

The most significant contribution of this work lies in highlighting the challenge of reproducibility. The performance of PaperQA was observed to be highly volatile, fluctuating based on API load during peak times—a phenomenon likely linked to the Mixture-of-Experts (MoE) architecture of models like GPT-4—and even exhibiting sensitivity to local hardware specifications. These external variables, which are outside the control of the user, introduce a troubling degree of variance and call into question whether these systems can be considered robustly superhuman if their performance cannot be consistently replicated.

Ultimately, while this project demonstrates that agentic RAG systems possess the raw capability to exceed human performance on specialized scientific question-answering tasks, it also serves as a warning on their practical readiness. The path to reliable deployment in scientific research depends not only on advancing the models themselves but also on establishing more stable infrastructure and transparent, standardized evaluation methodologies. Until then, 'superhuman' performance remains a brilliant but fragile achievement, heavily dependent on the specific implementation.

6 Future Work

The key issue is not whether language agents can achieve superhuman performance, but whether they can outperform humans consistently. The discussion talked about the implementation issues that caused flucatuations in performance (despite the agents being set to being determinisitic with 0 temperature). The best way to check this, would be to have an indication of peak times and to run the experiments again throughout these times, to get an indicator of average performance.

LitQA and PaperQA are biology-focused, but expanding the investigation to other scientific fields, where the process of science varies, could give a better indication of how good these RAG language agents are at performing science in general. Also, science is more than just sythesizing information. Science requires knowledge from disciplines such as mathematics, software engineering, and literature to create, test and publish new scientific hypotheses. Therefore, investigations into how Language agents can not only understand new information, but use it to write informed code and solve mathematical problems. Also, the creation of novel ideas is a topic that needs to be tested.

Acknowledgments

Acknowledge any funding, technical support, or other contributions.

References

[1] Non-determinism in GPT-4 is caused by Sparse MoE, August 2023. Section: posts.

[2] BeConvincible. Does ChatGPT get better / worse at different times of the day? (When under load.), November 2023.

[3] Sean Betts. Peering Inside GPT-4: Understanding Its Mixture of Experts (MoE) Architecture, July 2023.

[4] Hengxing Cai, Xiaochen Cai, Junhan Chang, Sihang Li, Lin Yao, Changxin Wang, Zhifeng Gao, Hongshuai Wang, Yongge Li, Mujie Lin, Shuwen Yang, Jiankun Wang, Mingjun Xu, Jin Huang, Xi Fang, Jiaxi Zhuang, Yuqi Yin, Yaqi Li, Changhong Chen, Zheng Cheng, Zifeng Zhao, Linfeng Zhang, and Guolin Ke. SciAssess: Benchmarking LLM Proficiency in Scientific Literature Analysis, October 2024. arXiv:2403.01976 [cs].

[5] Shailja Gupta, Rajesh Ranjan, and Surya Narayan Singh. A Comprehensive Survey of Retrieval-Augmented Generation (RAG): Evolution, Current Landscape and Future Directions, October 2024. arXiv:2410.12837 [cs].

[6] gwern. GPT-4 rumors: a Mixture-of-Experts w/8 GPT-3-220bs?, June 2023.

[7] Shanshan Han, Qifan Zhang, Yuhang Yao, Weizhao Jin, and Zhaozhuo Xu. LLM Multi-Agent Systems: Challenges and Open Problems, May 2025. arXiv:2402.03578 [cs].

[8] Jon M. Laurent, Joseph D. Janizek, Michael Ruzo, Michaela M. Hinks, Michael J. Hammerling, Siddharth Narayanan, Manvitha Ponnapati, Andrew D. White, and Samuel G. Rodriques.

LAB-Bench: Measuring Capabilities of Language Models for Biology Research, July 2024. arXiv:2407.10362 [cs].

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, April 2021. arXiv:2005.11401 [cs].

[10] Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. MathBench: Evaluating the Theory and Application Proficiency of LLMs with a Hierarchical Mathematics Benchmark, May 2024. arXiv:2405.12209 [cs].

[11] OpenAI, Ahmed El-Kishky, Alexander Wei, Andre Saraiva, Borys Minaiev, Daniel Selsam, David Dohan, Francis Song, Hunter Lightman, Ignasi Clavera, Jakub Pachocki, Jerry Tworek, Lorenz Kuhn, Lukasz Kaiser, Mark Chen, Max Schwarzer, Mostafa Rohaninejad, Nat McAleese, o3 contributors, Oleg Mürk, Rhythm Garg, Rui Shu, Szymon Sidor, Vineet Kosaraju, and Wenda Zhou. Competitive Programming with Large Reasoning Models, February 2025. arXiv:2502.06807 [cs].

[12] Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, Zekun Wang, Jian Yang, Zeyu Cui, Yang Fan, Yichang Zhang, Binyuan Hui, and Junyang Lin. CodeElo: Benchmarking Competition-level Code Generation of LLMs with Human-comparable Elo Ratings, January 2025. arXiv:2501.01257 [cs].

[13] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Scharli, and Denny Zhou. Large Language Models Can Be Easily Distracted by Irrelevant Context.

[14] Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D. Doan, and Chandan K. Reddy. LLM-SRBench: A New Benchmark for Scientific Equation Discovery with Large Language Models, April 2025. arXiv:2504.10415 [cs].

[15] Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela Hinks, Michael J. Hammerling, Manvitha Ponnapati, Samuel G. Rodriques, and Andrew D. White. Language agents achieve superhuman synthesis of scientific knowledge, September 2024. arXiv:2409.13740 [cs].

[16] Soumith Chintala [@soumithchintala]. i might have heard the same – I guess info like this is passed around but no one wants to say it out loud. GPT-4: 8 x 220B experts trained with different data/task distributions and 16-iter inference. Glad that Geohot said it out loud. Though, at this point, GPT-4 is, June 2023.

[17] swyx [@swyx]. @latentspacepod @realGeorgeHotz GPT4 is 8 x 220B params = 1.7 Trillion params https://t.co/DW4jrzFEn2 ok I wasn't sure how widely to spread the rumors on GPT-4 but it seems Soumith is also confirming the same so here's the quick clip! so yes, GPT4 is technically 10x the size of GPT3, and all the small https://t.co/m2YiaHGVs4, June 2023.

[18] Jialin Wang and Zhihua Duan. Agent AI with LangGraph: A Modular Framework for Enhancing Machine Translation Using Large Language Models, December 2024. arXiv:2412.03801 [cs].

[19] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, October 2023. arXiv:2308.08155 [cs].

[20] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. Hallucination is Inevitable: An Innate Limitation of Large Language Models, February 2025. arXiv:2401.11817 [cs].