Ai2 Scholar QA: Organized Literature Synthesis with Attribution

Amanpreet Singh* **Joseph Chee Chang*** **Chloe Anastasiades*** **Dany Haddad***
Aakanksha Naik **Amber Tanaka** **Angele Zamarron** **Cecile Nguyen** **Jena D. Hwang**
Jason Dunkleberger **Matt Latzke** **Smita Rao** **Jaron Lochner** **Rob Evans**
Rodney Kinney **Daniel S. Weld** **Doug Downey*** **Sergey Feldman***
Allen Institute for AI
{amanpreets, sergey}@allenai.org

Abstract

Retrieval-augmented generation is increasingly effective in answering scientific questions from literature, but many state-of-the-art systems are expensive and closed-source. We introduce Ai2 Scholar QA, a free online scientific question answering application. To facilitate research, we make our entire pipeline public: as a customizable open-source Python package[1] and interactive web app, along with paper indexes accessible through public APIs and downloadable datasets. We describe our system in detail and present experiments analyzing its key design decisions. In an evaluation on a recent scientific QA benchmark, we find that Ai2 Scholar QA outperforms competing systems.

qa.allen.ai
allenai/ai2-scholarqa-lib
Demo Video
Python Package

1 Introduction

Long-form scientific question answering systems use retrieval-augmented generation (RAG) (Lewis et al., 2020) over scientific literature to answer complex questions. These systems produce responses that bring together relevant insights from dozens of papers to help users rapidly learn about a body of scientific work. Examples are OpenScholar (Asai et al., 2024), Elicit, Consensus, and others §5.

Most of these systems are expensive to use and closed source, relying on models, workflows, and retrieval solutions not shared publicly. These issues create barriers for researchers who wish to study or build on the work. In response, we introduce Ai2 Scholar QA, a free-to-use scientific QA system (qa.allen.ai), and share our key components as open source software and public APIs.

Scholar QA follows a multi-stage pipeline (Figure 1) that starts by querying paper indexes: one from Semantic Scholar with over 100M abstracts, and a new index that we introduce in this work containing 11.7M full-text scientific papers. The pipeline then re-ranks the retrieved passages with a cross-encoder, and finally prompts a Large Language Model (LLM) to filter, cluster, and synthesize the passages into an answer. The final answer is presented to the user in a report with expandable sections of prose, bulleted lists, and tables. Claims in the answer are supported by citations, which can be clicked to reveal the cited paper's title and authors (with links to their corresponding Semantic Scholar pages), and in many cases relevant excerpt(s) from the paper, allowing for quick verification of the claim.

The system is based on open source code, enabling the community to reproduce and build on it. We release the code for our pipeline, prompting workflow and Web application. The retrieval indexes, including the new full text search index, are available as Semantic Scholar APIs and dataset downloads, and are continually updated with new articles (Kinney et al., 2023). Together, these resources can be combined with any generative LLM API to power a complete long-form scientific QA application. Our production system currently uses Anthropic's Claude 3.7 (Anthropic, 2024).

We present analyses that justify key design decisions in our architecture in §4. Our choice of retrieval models and configuration is informed by evaluation over a collection of real and synthetic user queries and accompanying passages judged for relevance by a LLM, both of which we release publicly. We compare Scholar QA's answers against several baselines, demonstrating that it achieves state-of-the-art performance on the ScholarQA-CS benchmark (Asai et al., 2024). Finally, we discuss the reception of Scholar QA by users. The strong majority (85%) of user feedback is positive, and the reported issues suggest important improvements for future work.

* Core contributors
[1]We use closed state-of-the-art LLMs.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 513–523

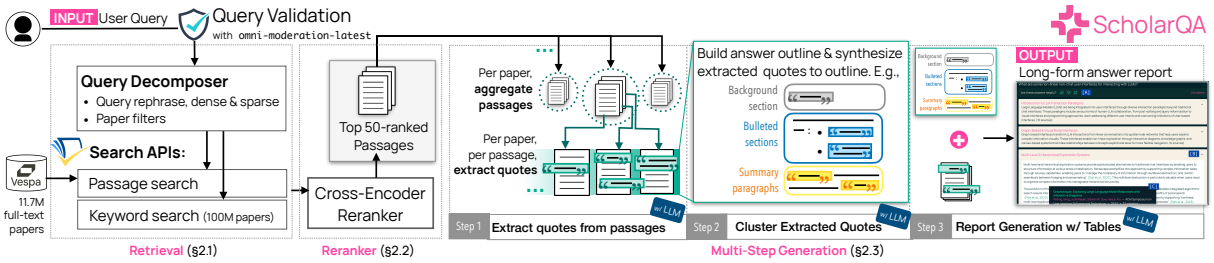

Figure 1: Scholar QA Pipeline Overview

2 Pipeline

The Scholar QA architecture (Figure 1) has three primary components: 1) retrieval to identify relevant passages from a corpus of scientific literature; 2) a neural cross-encoder that re-ranks the passages to select the most relevant top-k; and 3) multi-step LLM generation to process the passages into a comprehensive report. Next, we describe each component of the pipeline in detail.

Query Validation. Prior to processing a query, we employ OpenAI's `omni-moderation-latest`[2] model for safeguarding against potentially harmful content and return appropriate error messages.

2.1 Retrieval

We use the Semantic Scholar API (Kinney et al., 2023) for retrieval, specifically its endpoint for keyword search over paper abstracts, and our new endpoint for querying snippets from open-access papers. A query decomposer re-formulates the user query for each endpoint and retrieves up to 256 snippets and 20 abstracts. These texts are referred to as "passages" below.

Query Decomposer. The two retrieval endpoints differ in their effective query formats (one targets keyword and the other semantic queries) and filtering of results based on the user's preferences for paper metadata (paper year, venue, field of study). In our query decomposition step, an LLM is prompted to re-format the user query into paraphrases appropriate for each endpoint, and to extract the user's requested settings for the metadata filters. We use the outputs of this step for retrieval.

Search APIs. The Semantic Scholar keyword search API is described in Kinney et al. (2023). We introduce a new /snippet/search endpoint, which searches over a corpus of passages extracted from S2ORC (Lo et al., 2020), loaded into a Vespa cluster with papers and passages. Papers include metadata for filtering. Passages are derived from a paper's title, abstract, or body and can be filtered at the paper level. The index includes 11.7M full-text papers across the fields of study listed here, and a total of 285.6M passages.

Each passage is limited to 480 tokens and truncated at sentence and section boundaries where possible, having an overlap of one sentence (up to 64 tokens) with the preceding and following passages. Passage text is embedded with `mxbai-embed-large-v1` (Lee et al., 2024) with binary quantization, and placed into a dense (approximate nearest neighbor) index, as well as a traditional sparse keyword index.

We first retrieve a union of embedding and keyword-based matches, applying any specified filters. The filtered results are ranked with a weighted sum of embedding similarity and bm25 scores.

2.2 Reranking

The passages obtained from the retrieval step are subsequently passed to a neural re-ranker and the top 50 results are retained. The re-ranker is a cross-encoder that encodes both the query and a candidate document simultaneously and outputs a relevance score used to rank the documents. We selected `mxbai-rerank-large-v1` (Shakir et al., 2024) based on the results in §4.2 and host it on Modal with a single NVIDIA L40S GPU.

2.3 Multi-step Generation

The generation phase employs a three-step approach: first, the retrieved passages are processed to extract more precise quotes relevant to the query; second, the quotes are thematically clustered into separate sections appropriate for the answer; finally, a controlled generation process composes the final report one section at a time, synthesizing the quotes assigned to that section.

Quote extraction. Passages from the retrieval stage can be lengthy and may contain extraneous information not useful for answering the user query (Asai et al., 2023). The quote extraction stage aims

[2] https://platform.openai.com/docs/guides/moderation

to select only the most relevant quotes from the passages to improve the precision of the answer.

We instruct an LLM to extract verbatim quotes that directly contribute to answering the query (Slobodkin et al., 2024). As input to the extraction, we gather all passages from the re-ranker for a given paper, and concatenate these to the abstract of the paper. This aggregation helps create a richer context conducive to extracting relevant quotes. The LLM processes each paper's content independently and returns the selected quotes separated by ellipses. If the entire paper context is deemed irrelevant, it is discarded from further processing.

Answer Outline and Clustering. For generating a comprehensive research report, the effective organization of reference materials is essential for its overall coherence. We propose a thematic outline framework where the answer is divided into sections representing topics, and the reference quotes are assigned to these topics. This mapping allows the system to selectively focus only on the pertinent subset of quotes when synthesizing a section.

First, the LLM is instructed to generate a list of themes in logical order and the appropriate synthesis format for each theme, independent of the quotes from the previous step. The first section is always an introduction or background to provide the user the basics for understanding the answer. The format of each section can be either a paragraph or a bulleted list, serving different information needs. Paragraphs convey nuanced summaries from multiple papers, while bulleted lists enumerate related papers (e.g., models, datasets, or interactive systems). These list are also the catalyst for generating the comparison tables (see §2.3). Following this, the sections are assigned 0 or more quotes. In case no quote is assigned to a section, it is generated completely from the LLM weights.

Report Generation. With the answer outline in place, each section of the report is synthesized serially conditioned on the query, reference sources, and the sections prior to it. The LLM is also instructed to generate a TLDR for each section. The references are either the quotes assigned to the section or abstracts of papers that are cited within these quotes. This citation following method allows the LLM to condition on and cite foundational sources which are not uncovered in retrieval. The LLM is instructed to cite the sources for each claim in the generated section text and cite generations from its parameters as `LLM Memory`.

Paper Comparison Table Generation. Since bulleted list sections typically include closely related papers (e.g., different *datasets*), we additionally generate tables that compare and contrast all papers cited in that section using common aspects (e.g., *size* and *annotation method*). This pipeline is detailed in Newman et al. (2024). At a high level, the inputs are the query to Scholar QA, the section title, and the abstracts of all papers cited in the section. An LLM first produces a set of common aspects (columns) to compare papers (rows). Each cell (paper-aspect pair) is filled with a value using the full-text of the paper. Finally, as not all aspects are applicable to every paper (e.g., one paper might not be about a dataset), we filter out columns and rows with a high proportion of missing values. Figure 3 [A] shows an expanded table in Scholar QA where related papers from a section are compared across a set of common aspects ([B]).

3 Scholar QA: Interface and Source Code

Scholar QA is open-sourced as an extensible Python package (`ai2-scholar-qa`) and a Typescript and React-based interactive web application. The LLM functionality of Scholar QA is implemented with `litellm`, which supports swapping a variety of models using your own keys. Thus, the community can build upon Scholar QA and easily visualize the results (examples in Appendix A). Below we describe the user experience of the demo.[3]

Progress and Section Streaming. High system latency can hinder usability. On average, Scholar QA produces a full report in 2.5 minutes (N=500, σ=70s), which is comparable to modern LLM-based research tools. To further improve usability, the following designs were used: 1) Displaying detailed real-time progress of the system (Nielsen, 1994) so users can examine the number of papers, passages, and sections being processed. 2) Presenting each section as soon as it is generated, so users can begin browsing the first section in 50 seconds (N=500, σ=24s) post issuing a query (Appendix H).

Expandable Sections. By default, sections are collapsed showing only their titles, TLDR summaries, and number of cited sources. This gives users a gist of the information included in the report (Figure 2 [A]). Users can then click on the title of a section they wish to read to expand it ([B]).

[3]Our production system has a few additional features like downloadable reports, login and links to other Ai2 systems.

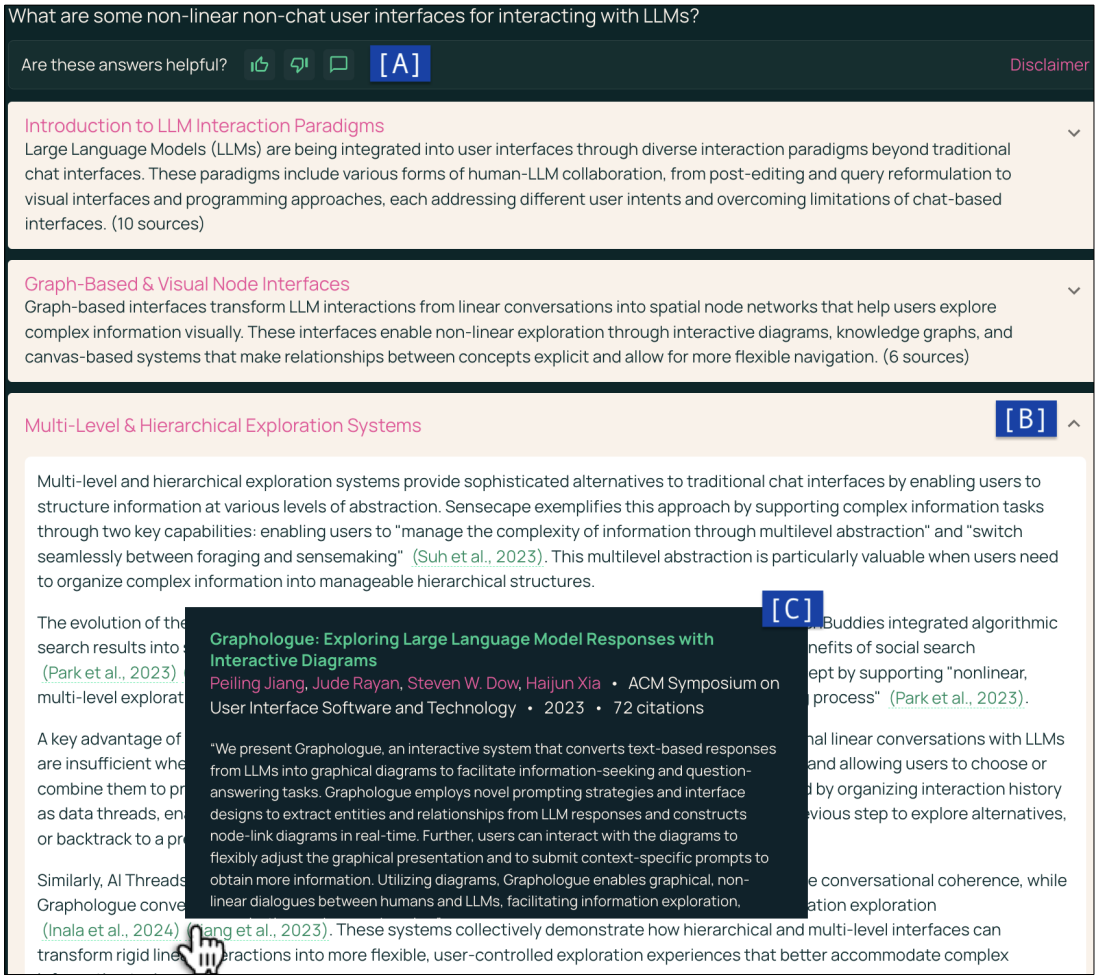

Figure 2: Multi-section [B] report generated by Scholar QA. References are linked to supporting excerpts [C]. Thumbs and free text feedback are collected for the full report [A], and also for each section and inline table.

References and Evidence Excerpts. To verify the claims in the report, users can click on the inline citations (Figure 2 [C]) or the pink excerpt icon in the inline table cells (Figure 3 [C]) to bring up a popup paper card. From the paper card, they can see the relevant excerpts used during the generation or click on the title to open the paper directly.

User Feedback Collection. We collect thumbs up/down or textual feedback for the whole report (Figure 2 [A]) and at each section and inline table.

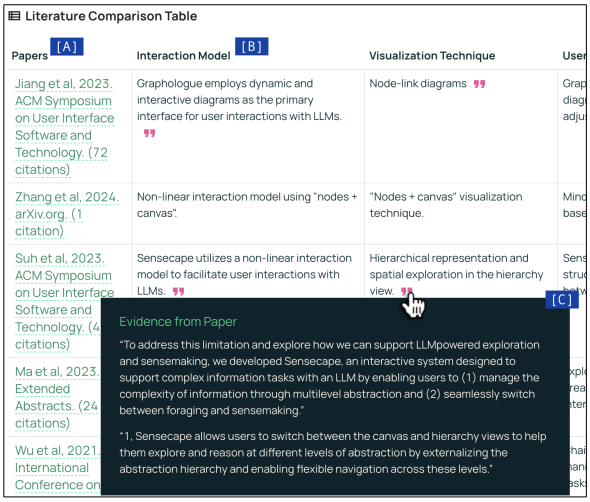

Figure 3: Inline tables compare papers [A] with common aspects [B] with values linked to supporting excerpts from the papers [C].

4 Evaluation

4.1 Retrieval

We tuned our retrieval setup by optimizing ranking over a dev set of 500 synthetic queries (see Appendix C) and the top 1000 passages for each based on GIST embedding distance (Solatorio, 2024). We generated binary relevance labels with `gpt-4-turbo` (see Appendix B for the prompt), which were found to have 80% agreement with

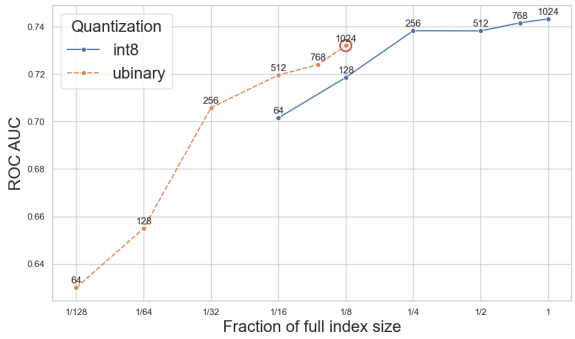

Figure 4: Embedding ranking performance for various compression methods and matryoshka cutoffs. The x-axis indicates the size of the vector index based relative to using `int8` quantization and the full embedding size. The red circle indicates the selected configuration. Embedding size is notated next to each point.

human annotators on a sample of 100 queries.

Pipeline Tuning. We optimized several aspects of retrieval over this dev set: embedding model selection and quantization method for it, the components and weights in the final ensemble, and (when relevant) the target Matryoshka dimension for the embeddings (Kusupati et al., 2024).

We experimented with medium sized embedding models based on top performers on the retriever and ranking tasks of the MTEB (Muennighoff et al., 2022) leaderboard on HuggingFace. Table 4 in Appendix D lists our candidate models. The `mxbai-embed-large-v1` (Lee et al., 2024) embeddings performed best over our dev set. Figure 4 validates our choice of quantization method and target Matryoshka dimension for these embeddings. We chose `ubinary` quantization with no Matryoshka truncation, (indicated by a red circle on the plot) since it satisfied our storage constraints without a large drop in performance. We experimented with ensembling `SparseEmbed` (Kong et al., 2023), embedding cosine similarity, BM25, and chose the latter two (weight split of $(0.6, 0.4)$ respectively) based on the results (See Appendix E). The BM25 scores are normalized with min-max scaling before computing the ensemble score.

4.2 Reranking

We chose the re-ranker based on evaluation over a mixture of real scientific questions from the Stack Exchange Computer Science, Math, and Statistics communities, real research queries written by the authors and their colleagues, and synthetic ones generated by fine-tuning GPT-4o-mini over questions from the ScholarQA-CS dataset (Asai et al., 2024). For a given query, passages are retrieved and then awarded a relevance score in the range 0-3 with GPT-4o. We experiment with multiple state-of-the-art re-rankers (Chen et al., 2024; Shakir et al., 2024; Asai et al., 2024), and, as shown in Table 2, `mxbai-rerank-large-v1` gives the best results across the board (even outperforming its v2 model on our task). To reduce latency for deployment, we implemented optimizations like Pytorch model compilation. We release the evaluation data consisting of 2,426 queries and 225,618 passages.

Model (Size)	Latency (sec/query)	nDCG @10	mRR
bge-reranker-v2-m3 (568M)	0.14	0.913	0.973
akariasai/ranker_large (568M)	0.14	0.906	0.970
jina-reranker-v2-base (278M)	**0.06**	0.907	0.972
mxbai-rerank-large-v1 (435M)	0.46	**0.927**	**0.975**
mxbai-rerank-base-v1 (184M)	0.19	0.919	0.974
mxbai-rerank-xsmall-v1 (70M)	0.11	0.911	0.970
mxbai-rerank-base-v2 (0.5B)	0.40	0.918	0.974
mxbai-rerank-large-v2 (1.5B)	0.70	0.911	0.975

Table 1: Cross encoder re-ranker results on our dataset of GPT-4o labels. The best results are **highlighted**.

4.3 Generation

We evaluate the final output of Scholar QA on the ScholarQA-CS dataset which consists of expert-annotated rubrics for 100 Computer Science research questions. The question-specific expert rubrics account for 60% of the final score, while the rest is computed based on global metrics of length, expertise and citations. We use GPT-4o (Hurst et al., 2024) as a judge with the utility provided by Asai et al. (2024) for automatic evaluation and compare against several baselines.

As shown in Table 2, our system outperforms popular LLMs: Llama 3.1 (Dubey et al., 2024), GPT 4.1 and Claude Sonnet 3.7 (Anthropic, 2024). It even outperforms reasoning models such as Sonnet 3.7 Thinking (Anthropic, 2025), o1-mini (OpenAI, 2024b) and o3-mini (Zhang et al., 2025) overall on the Scholar QA-CS benchmark. This setup lacks any retrieval so the models generate the responses completely from parametric memory. The benchmark rewards attribution and supporting evidence as a measure of trust in the system, so these models score lower overall. The reasoning based models perform better than our system on the rubrics score, which suggests that they may be superior backbones for our system. However, due to the additional reasoning tokens, these models are more expensive and also significantly increase latency.

For contemporary QA systems, we compare against OpenScholar with GPT-4o[4], PaperQA2 (Skarlinski et al., 2024), Perplexity's `Sonar Deep Research` and STORM (Shao et al., 2024a). PaperQA2 did not release their retrieval corpus, so we substitute it with our retrieval pipeline for a fair comparison. Scholar QA obtains the best scores both on rubrics and overall, with the variant using Claude 3.7 Sonnet as the backbone scoring 2.4 points higher than STORM. For these QA systems, we also evaluate the attribution quality based on ALCE (Gao et al., 2023), which proposes entailment between claims and evidence to compute citation precision and recall. Again, we use GPT-4o as a judge to predict entailment (See Appendix F for the prompt) and treat each sentence in a response as a claim. Even with a report spanning multiple sections where all the sentences might not be cited, Scholar QA comes out far ahead of the other QA systems. Due to a lack of retrieval, this evaluation was not conducted when the LLMs are simply prompted to generate a response from memory. An interesting discovery from our analysis was that with an updated version of GPT-4o (i.e. `gpt-4o-2024-11-20`) as the judge, the scores are inflated compared to using `gpt-4o-2024-08-06`, even though the relative rankings are consistent (See Appendix J). For parity with Asai et al. (2023), we report the rubrics and citation scores with the older and newer model as the judge, respectively.

During our initial experiments, we restricted ScholarQA to only summarize the insights conditioned on the quotes extracted from retrieved passages. However, in cases where the retrieved passages were not relevant enough, the system failed to answer the question in favor of just discussing the information in the quotes. Moreover, for over 30% of instances in ScholarQA-CS, the rubrics require background information, even though the question might not. So, we updated our system LLM prompts to – a) Generate section text from memory if there is a lack of relevant retrieved passages and cite as LLM Memory and b) generate the first section as a background or introduction for the rest of the answer. The results reported here are obtained post these changes.

To finalize the backbone LLM for the production web application we conducted an anonymized pairwise comparison among the authors of this work. We compare Claude 3.7 against 3.5. Out of 18 comparisons, Claude 3.7 Sonnet was the overwhelming favorite with 17 wins, reinforcing our hypothesis that (with no other changes) our system improves with newer and better backbone LLMs.

[4]Our results are not identical to Asai et al. (2024) due to variance across LLM-as-a-judge runs. Their reported total score for OS-GPT-4o is 57.7. We re-ran the evaluation in order to obtain rubrics only scores, which they did not report.

Model	Score		Model	Score		
	Rubrics	Total		Rubrics	Total	Cite
LLM Prompting (No Retrieval)			*QA Systems*			
Llama 3.1-8B	48.8	47.3	SQA-Claude 3.7 S	58.0	**61.9**	48.1
Llama 3.1-70B	52.4	48.6	SQA-Claude 3.5 S	52.6	61.3	**52.1**
Claude 3.5 S	50.4	46.6	OS-GPT-4o	49.3	53.5	25.9
Claude 3.7 S	61.5	55.9	PaperQA2	38.7	51.4	25.3
+Thinking	62.7	55.7	Perplex. Sonar DR	38.7	52.8	25.2
GPT-4.1	**63.2**	56.2	STORM	54.2	59.5	40.2
o1-mini	62.3	55.5				
o3-mini	60.6	50.2				

Table 2: Evaluation results on ScholarQA-CS benchmark. System responses are either generated by simply prompting LLMs with the questions or by issuing the queries to RAG based QA systems. Expert annotated rubrics only scores are reported in addition to the overall total. The overall best results are **highlighted** and best results within a category are underlined. SQA: Ai2 Scholar QA, OS: Open Scholar, S: Sonnet, Claude 3.5 S: `claude-3-5-sonnet-20241022`.

4.4 Real-world Usage and User Feedback

We have publicly deployed Scholar QA for 9 weeks, and received 30.2k questions from 8,219 unique visitors. On average, each response is about 2.4k words and costs $0.50 to produce. We observed 1,075 monthly repeated users who had issued queries on two distinct days over the course of a 30 day window. We analyze the user query types and the most prominent themes were *deep-dive* into specific research topics (15k) and *comparative analysis* of specific prior work (5k) (detailed distribution in Appendix I). A total of 2,433 thumbs feedback were submitted (Figure 2 [A]) and 85% were positive. These suggests real-world users benefited from using Scholar QA.

For insight into the failure modes, we manually examined the 383 instances of neutral/negative free-form feedback. Table 3 lists the feedback types we identified along with their counts as of May 2025 (example feedback in Appendix G). We hypothesize that follow-up questions may help address insufficient answer detail and cases with a lack of retrieved documents, while improved retrieval may help address incomplete or incorrect references and off-topic responses.

Category	Count
Incorrect or Missing References	126
Off-topic or Misunderstood Query	113
Request for More Detail or Specificity	289
General Feedback on Quality	149
Language or Format Issues	78

Table 3: Feedback Categories and Counts

5 Related Work

Scientific Question Answering. Answering scientific questions involves navigating scholarly sources and accurately retrieving and synthesizing them. Recently, OpenScholar (Asai et al., 2024) introduced a retrieval-augmented model designed explicitly for scientific literature synthesis with citation-supported responses with significant improvement in accuracy and reduced citation hallucination. Scholar QA extends its capabilities by leveraging the latest state-of-the-art LLMs and an open source generation pipeline that filters literature into precise quotes and produces thematically organized and detailed answers. STORM (Shao et al., 2024b) synthesizes comprehensive, Wikipedia-like articles, a distinct task from long-form scientific question answering. Other works have focused on literature review synthesis: LitLLM (Agarwal et al., 2024), which like Scholar QA uses a structured planning-and-generation pipeline similar, and SurveyForge (Yan et al., 2025), which outlines heuristics before generation. Their code was not available at the time of our evaluation. Zhou et al. (2025) present a survey categorizing AI-driven research support systems across various stages of the scientific process, including literature synthesis.

Commercial Tools for Scientific QA. Commercial RAG tools have emerged to facilitate research specifically tailored for scientific literature, such as Consensus (Consensus, 2024), which synthesizes findings from research papers, Scite (Scite, 2024), which evaluates claims by analyzing citation contexts, and Elicit (Elicit, 2024), which supports structured scientific literature reviews. Other general-purpose tools also support scientific inquiries: Perplexity (Perplexity, 2024), You.com (You.com, 2024), OpenAI Deep Research (OpenAI, 2024a) and Gemini Deep Research (DeepMind, 2024). Although these platforms leverage advanced retrieval and generation capabilities to facilitate literature reviews and deliver rapid insights, they can be too expensive for widespread academic use and typically lack transparency regarding their pipelines. In contrast, Scholar QA is free with open sourced code and access to search APIs that enable the research community to build upon it.

6 Conclusion

We present Ai2 Scholar QA, a freely-available long-form literature synthesis system that generates reports for complex scientific questions. We release key components as open source code and public APIs, and report experiments analyzing design decisions and demonstrate state-of-the-art results.

Limitations

Supplementing the user feedback discussed in subsection 4.4, we would like to outline some limitations of our system and evaluation and our plans to mitigate them as part of fuure work:

(i) Ai2 Scholar QA uses proprietary and closed-source LLM as the backbone for our production pipeline. As shown in Table 2, open source models lag behind the proprietary models in our evaluation. However, we are actively experimenting with open-sourced LLMs to replace the closed ones partially or completely in the pipeline. The open-sourced models will be specifically trained to do well on long-form scientific question answering and each of the sub-tasks in our multi-step generation. Further, our code is open-sourced and can easily be used with potentially any available LLM api provider supported by litellm.

(ii) We evaluate the answers generated by Scholar QA and compare against other systems on ScholarQA-CS dataset in subsection 4.3. Even though the answer rubrics are collected via human annotation, the evaluation is only limited to questions in the Computer Science domain and further relies completely on an LLM as the evaluator. In ongoing work, we are investigating more accurate benchmarks for evaluating long form scientific answers. Our approach uses real queries posed by users to Scholar QA, and human preference labels over answers from multiple systems in not just Computer Science, but Biomedicine and other scientific domains. These labels can serve as not only for evaluation, but also as training signals for models.

Acknowledgments

We would like to thank the anonymous reviewers for helpful comments, suggestions and feedback on the manuscript. We would also like to acknowledge the Ai2 ScholarQA users for providing constructive feedback that helped us improve the system. Finally, we thank David Albright for helping with the demo video, the Ai2 communications team for their help with user outreach, and Ai2 engineers and researchers for their help with user testing before launch.

References

Shubham Agarwal, Gaurav Sahu, Abhay Puri, Issam Hadj Laradji, Krishnamurthy Dj Dvijotham, Jason Stanley, Laurent Charlin, and Christopher Pal. 2024. Litllms, llms for literature review: Are we there yet?

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku.

Anthropic. 2025. Claude 3.7 sonnet system card.

Akari Asai, Jacqueline He, Rulin Shao, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Mike D'Arcy, David Wadden, Matt Latzke, Minyang Tian, Pan Ji, Shengyan Liu, Hao Tong, Bohao Wu, Yanyu Xiong, Luke S. Zettlemoyer, and 6 others. 2024. Openscholar: Synthesizing scientific literature with retrieval-augmented lms. *ArXiv*, abs/2411.14199.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *ArXiv*, abs/2310.11511.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2402.03216.

Consensus. 2024. Consensus – ai for research. Accessed: 2025-03-28.

Google DeepMind. 2024. Gemini – deep research mode. Accessed: 2025-03-28.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony S. Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 510 others. 2024. The llama 3 herd of models. *ArXiv*, abs/2407.21783.

Elicit. 2024. Elicit – the ai research assistant. Accessed: 2025-03-28.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Conference on Empirical Methods in Natural Language Processing*.

OpenAI Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mkadry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alexander Kirillov, Alex Nichol, Alex Paino, and 397 others. 2024. Gpt-4o system card. *ArXiv*, abs/2410.21276.

Rodney Michael Kinney, Chloe Anastasiades, Russell Authur, Iz Beltagy, Jonathan Bragg, Alexandra Buraczynski, Isabel Cachola, Stefan Candra, Yoganand Chandrasekhar, Arman Cohan, Miles Crawford, Doug Downey, Jason Dunkelberger, Oren Etzioni, Rob Evans, Sergey Feldman, Joseph Gorney, David W. Graham, F.Q. Hu, and 29 others. 2023. The semantic scholar open data platform. *ArXiv*, abs/2301.10140.

Weize Kong, Jeffrey M. Dudek, Cheng Li, Mingyang Zhang, and Michael Bendersky. 2023. Sparseembed: Learning sparse lexical representations with contextual embeddings for retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page 2399–2403. ACM.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2024. Matryoshka representation learning. *Preprint*, arXiv:2205.13147.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. 2024. Open source strikes bread - new fluffy embeddings model.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *ArXiv*, abs/2005.11401.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Michael Kinney, and Daniel S. Weld. 2020. S2orc: The semantic scholar open research corpus. In *Annual Meeting of the Association for Computational Linguistics*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Benjamin Newman, Yoonjoo Lee, Aakanksha Naik, Pao Siangliulue, Raymond Fok, Juho Kim, Daniel S. Weld, Joseph Chee Chang, and Kyle Lo. 2024. Arxivdigestables: Synthesizing scientific literature into tables using language models. In *Conference on Empirical Methods in Natural Language Processing*.

Jakob Nielsen. 1994. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 152–158.

OpenAI. 2024a. Chatgpt – deep research mode. Accessed: 2025-03-28.

OpenAI. 2024b. Openai o1 system card.

Perplexity. 2024. Perplexity ai – ask anything. Accessed: 2025-03-28.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Scite. 2024. Scite – smart citations for research. Accessed: 2025-03-28.

Aamir Shakir, Darius Koenig, Julius Lipp, and Sean Lee. 2024. Boost your search with the crispy mixedbread rerank models.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024a. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, Mexico City, Mexico. Association for Computational Linguistics.

Yijia Shao, Yucheng Jiang, Theodore A. Kanell, Peter Xu, Omar Khattab, and Monica S. Lam. 2024b. Assisting in writing wikipedia-like articles from scratch with large language models. *Preprint*, arXiv:2402.14207.

Michael D. Skarlinski, Sam Cox, Jon M. Laurent, James D. Braza, Michaela M. Hinks, Michael J Hammerling, Manvitha Ponnapati, Samuel G. Rodriques, and Andrew D. White. 2024. Language agents achieve superhuman synthesis of scientific knowledge. *ArXiv*, abs/2409.13740.

Aviv Slobodkin, Eran Hirsch, Arie Cattan, Tal Schuster, and Ido Dagan. 2024. Attribute first, then generate: Locally-attributable grounded text generation. In *Annual Meeting of the Association for Computational Linguistics*.

Aivin V. Solatorio. 2024. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *ArXiv*, abs/2402.16829.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Xiangchao Yan, Shiyang Feng, Jiakang Yuan, Renqiu Xia, Bin Wang, Bo Zhang, and Lei Bai. 2025. Surveyforge: On the outline heuristics, memory-driven generation, and multi-dimensional evaluation for automated survey writing.

You.com. 2024. You.com – personalized ai search. Accessed: 2025-03-28.

Brian Zhang, Eric Mitchell, Hongyu Ren, Kevin Lu, Max Schwarzer, Michelle Pokrass, Shengjia Zhao, Ted Sanders, Adam Kalai, Alexandre Passos, Benjamin Sokolowsky, Elaine Ya Le, Erik Ritter, Hao Sheng, Hanson Wang, Ilya Kostrikov, James Lee, Johannes Ferstad, Michael Lampe, and 93 others. 2025. Openai o3-mini system card.

Zekun Zhou, Xiaocheng Feng, Lei Huang, Xiachong Feng, Ziyun Song, Ruihan Chen, Liang Zhao, Weitao Ma, Yuxuan Gu, Baoxin Wang, Dayong Wu, Guoping Hu, Ting Liu, and Bing Qin. 2025. From hypothesis to publication: A comprehensive survey of ai-driven research support systems.

A Python Package Usage

Figure 5 shows a minimal example of running the system pipeline with the ai2-scholar-qa python package and how every component can be extended or modified as the users see fit.

```python
from scholarqa.rag.reranker.reranker_base import CrossEncoderScores
from scholarqa.rag.retrieval import PaperFinderWithReranker
from scholarqa.rag.retriever_base import FullTextRetriever
from scholarqa import ScholarQA

CLAUDE_SONNET_3_7 = "anthropic/claude-3-7-sonnet-20250219"
#Extends the scholarqa.rag.retrieval.AbstractRetriever class
retriever = FullTextRetriever(n_retrieval=256, n_keyword_srch=20)
#Extends the scholarqa.rag.reranker.reranker_base.AbstractReranker class
reranker = CrossEncoderScores("mixedbread-ai/mxbai-rerank-large-v1")
#Wrapper class for retrieval
paper_finder = PaperFinderWithReranker(retriever, reranker, n_rerank=50,
                context_threshold=0.5)
#Scholar QA wrapper with the MultiStepQAPipeline integrated
scholar_qa = ScholarQA(paper_finder, llm_model=CLAUDE_SONNET_3_7)
print(scholar_qa.answer_query("Which is the 9th planet in our solar system?"))

#Custom MultiStepQAPipeline class/steps
from scholarqa.rag.multi_step_qa_pipeline import MultiStepQAPipeline
mqa_pipeline = MultiStepQAPipeline(llm_model=CLAUDE_SONNET_3_7)
paper_quotes = mqa_pipeline.step_select_quotes(query,...)#Quote Extraction
plan = mqa_pipeline.step_clustering(query, paper_quotes,...)#Outline and Clustering
#Section Generation
response = list(mqa_pipeline.generate_iterative_summary(query, paper_quotes, plan,...))
```

Figure 5: `ai2-scholar-qa` usage example

B Document Relevance Prompt

We used the following prompt to obtain binary relevance labels, which agreed with human annotators 80% of the time:

```
If any part of the following text is relevant to the following question, then return 1, otherwise return 0. Non-english results are not relevant, results which are primarily tables are not relevant.
```

C Retrieval Tuning Query Generation

Queries for the dev set were obtained from three internal sources of human research questions, and a set of LLM generations. We experimented with several methods for constructing the synthetic LLM questions. Our approach was to generate questions similar to those asked by real users by prompting the LLM to output: (1) a question based on paragraphs retrieved from the corpus, and (2) a "more general" version of the first question. We only use the "more general" set since they were more similar to real user queries.

D Embedding Models for Retrieval

We experimented with multiple top embedding models from the MTEB leader board to optimize retrieval for our system. These are outlined in Table 4.

HuggingFace embedding model name
`Snowflake/snowflake-arctic-embed-m`[5]
`sentence-transformers/all-mpnet-base-v2` (Reimers and Gurevych, 2019)
`avsolatorio/GIST-Embedding-v0` (Solatorio, 2024)
`Snowflake/snowflake-arctic-embed-m-long`[6]
`intfloat/e5-base-v2` (Wang et al., 2022)
`mixedbread-ai/mxbai-embed-large-v1` (Lee et al., 2024)
`jinaai/jina-embeddings-v3` (Sturua et al., 2024)

Table 4: Embedding Models to optimize retrieval

E Retrieval Ensemble Experiments

Figure 6 shows results of our ensembling experiments for the full-text retrieval index. SparseEmbed introduces an overhead with minimal performance gains, so we picked an ensemble of embedding similarity and BM25 as our final ranking metric.

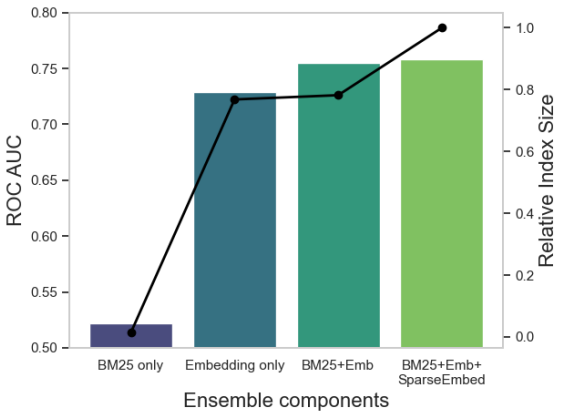

Figure 6: Ranking performance for various ensembles with relative size of the index required. Excluding SparseEmbed reduces the index size by 20% without a significant drop in ranking performance.

F Prompt for Evaluating Attribution

```
As an Attribution Validator, your task is to verify whether a given reference can support the given claim. A claim can be either a plain sentence or a question followed by its answer. Specifically, your response should clearly indicate the relationship: Attributable, Contradictory or Extrapolatory. A contradictory error occurs when you can infer that the answer contradicts the fact presented in the context, while an extrapolatory error means that you cannot infer the correctness of the answer based on the information provided in the context. Output your response as a json with only a single key "output" and a value of one among - ("Attributable", "Contradictory",
```

```
"Extrapolatory").
Claim: claim
Reference: ref_excerpt
```

G User Feedback Examples

Table 5 lists some examples of the user complaints for Scholar QA reports.

Feedback
The structure is good, but the articles you choose are not from top journals.
The first citation says that *rabbits* can obtain cholesterol from diet, not rats.
These provide a lot of general information about the topic, but nothing here actually addresses the central question I asked.
The answer did not address the 'MOBILIZATION' techniques at all! The answer is wrong because it addressed Exercise therapy!
They address the general setting, but not the specific question I asked.
It's only analysing on SASAF model, but there are more.

Table 5: Example Feedback on Research Issues

H Progress Updates and Report Sections

Figure 7 demonstrates how we display in real-time the progress of the system during generation. This included number of papers and passages the were processed in each step, as well as the outline as it is being generated. Each section appears as soon as it is generated, so users can begin browsing the first sections.

Figure 7: Progress indication and section streaming.

I Query Type Analysis

To analyze the types of questions users are asking, we use an LLM to categorize the queries. The most prominent types were comprehensive *deep-dive* into a specific research topic (15k) and *comparative analysis* of prior work (5k). Other themes such as factoid QA or specific methods, datasets accounted for fewer queries.

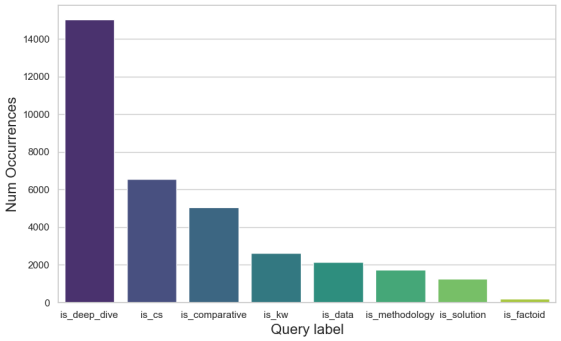

Figure 8: Distribution of different question types submitted to Scholar QA deployed Web application.

J Generation Results with updated GPT-4o

Table 6 shows results on ScholarQA-CS with `gpt-4o-2024-11-20` as the LLM judge. These results can be contrasted with the first two columns in Table 2 which are obtained with `gpt-4o-2024-08-06` as the judge. Even though the absolute scores are inflated compared to Table 2, the relative rankings are about the same with Scholar QA getting the best overall score.

Model	Score		Model	Score	
	Rubrics	Total		Rubrics	Total
LLM Prompting (No Retrieval)			*QA Systems*		
Llama 3.1-8B	51.8	48.2	SQA-Claude 3.7 S	67.3	**67.2**
Llama 3.1-70B	57.0	51.2	SQA-Claude 3.5 S	61.3	67.1
Claude 3.5 S	57.8	51.3	OS-GPT-4o	54.9	59.9
Claude 3.7 S	68.4	60.8	PaperQA2	43.8	54.1
+Thinking	68.3	58.7	Perplex. Sonar DR	43.9	56.0
GPT-4.1	**69.3**	61.8	STORM	59.2	64.7
o1-mini	69.1	61.3			
o3-mini	68.5	55.9			

Table 6: Evaluation results on ScholarQA-CS benchmark with `gpt-4o-2024-11-20` as the judge. System responses are either generated by simply prompting LLMs with the questions or by issuing the queries to RAG based QA systems. Expert annotated rubrics only scores are reported in addition to the overall total. The overall best results are **highlighted** and best results within a category are underlined. SQA: Ai2 Scholar QA, OS: Open Scholar, S: Sonnet, Claude 3.5 S: `claude-3-5-sonnet-20241022`.