Literature discovery with natural language queries

Anna Kiepura[†], **Jessica Lam**[†], **Nianlong Gu**[‡]
Richard H.R. Hahnloser[†]
[†]Institute of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland
{akiepura, lamjessica, rich}@ini.ethz.ch
[‡]Linguistic Research Infrastructure, University of Zurich, Switzerland
nianlong.gu@uzh.ch

Abstract

Literature discovery is a critical component of scientific research. Modern discovery systems leveraging Large Language Models (LLMs) are increasingly adopted for their ability to process natural language queries (NLQs). To assess the robustness of such systems, we compile two NLQ datasets and submit them to nine widely used discovery platforms. Our findings reveal that LLM-based search engines struggle with precisely formulated queries, often producing numerous false positives. However, precision improves when LLMs are used not for direct retrieval but to convert NLQs into structured keyword-based queries. As a result, hybrid systems that integrate both LLM-driven and keyword-based approaches outperform purely keyword-based or purely LLM-based discovery methods.

1 Introduction

Scientific research heavily relies on the ability to discover and assimilate relevant literature (Patel and Patel, 2019). Traditional literature search methods, whether through publisher-specific databases (e.g., *Nature*[1]) or generic academic search engines (e.g., *Google Scholar*[2]), primarily use keyword-based queries processed via inverted indexes and ranking algorithms such as BM25 (Robertson et al., 2009). While these conventional approaches are effective, they often struggle with nuanced, concept-driven queries.

Recent advancements in artificial intelligence, particularly the rise of Large Language Models (LLMs) and LLM-powered chatbots (e.g. *Consensus*[3]), have enabled a more intuitive and context-aware search experience. However, despite their convenience, LLM-driven retrieval systems lack formal guarantees of accuracy (Liu et al., 2023). This limitation can lead to erroneous search results, including false positives (retrieving irrelevant papers) and false negatives (overlooking relevant papers), potentially impacting the reliability of literature discovery. To systematically assess these challenges, we conduct an evaluation of various literature search engines using natural language queries (NLQs).

We find that existing platforms are not equipped to handle NLQs, with most papers retrieved being incorrect, but that using LLMs to parse NLQs into structured queries interpretable by these platforms highly boosts retrieval performance. Our contributions are:

1. We introduce two manually curated datasets[4] designed for systematic evaluation of literature discovery platforms on NLQs.
2. We benchmark the performance of nine popular literature discovery platforms on our datasets.
3. We investigate the ability of LLMs to transform NLQs into structured formats and analyze their impact on retrieval effectiveness.

2 Related work

Literature discovery, or the task of finding relevant papers (either to cite in a given sentence (Jeong et al., 2019; Kieu et al., 2020; Gu et al., 2022; Nogueira et al., 2020)) or to answer an input question (Menick et al., 2022; Gao et al., 2023; Dehghan et al., 2024)), has been a long-standing research focus in the realm of scientific document processing. Sun et al. (2024) and Ajith et al. (2024) showed that well-instructed LLMs outperform the more typical, nearest-neighbour based methods in re-ranking papers on relevance to the input query, in part due to the LLM ability to generalise across synonyms and imprecise queries.

However, because generation-based search en-

[1] nature.com/search/advanced
[2] scholar.google.com
[3] consensus.app
[4] https://github.com/annamkiepura/lit_discovery

Proceedings of the Fifth Workshop on Scholarly Document Processing (SDP 2025), pages 83–95

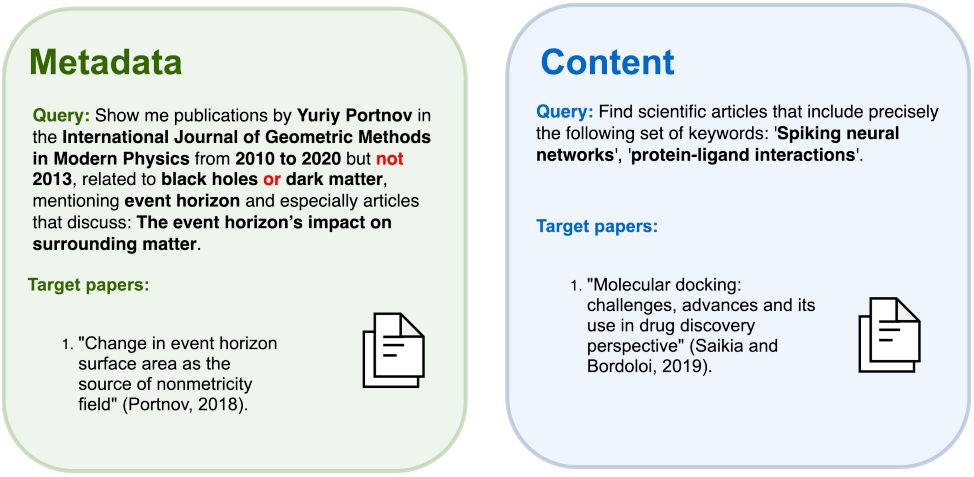

Figure 1: Example queries from the **Metadata** and **Content** datasets with the corresponding target papers (Portnov, 2018; Saikia and Bordoloi, 2019).

gines are prone to hallucination (Liu et al., 2023), grounding LLM-powered literature discovery platforms by referencing a paper database remains key. Retrieving information from most databases involves submitting structured queries that are interpretable by said databases. To reduce the need for learning about the structure accepted by each database, much research has gone into automatically translating NLQs into structured queries (Zhong et al., 2018). A typical approach was to manually craft a mapping table between mentioned keywords and database properties (Montgomery et al., 2020), but these tables are often too rigid to effectively handle query ambiguity and complexity. In contrast, recent works have successfully used LLMs for this mapping problem (Lei et al., 2024), and in this work, we explore whether this effectiveness extends to the specific context of precise NLQs for literature discovery.

3 Methods

A key principle of effective retrieval systems is to provide users with fine-grained control over retrieval behaviour (Schleith et al., 2022; Kandula et al., 2024). In literature discovery, this control would include enabling researchers to specify authors, venue, publication year, and topics of interest. Alternatively, researchers might seek papers on multidisciplinary topics defined by one or several keywords that are rarely used together. Such complex queries are well-suited to classical search engines but may pose challenges for LLMs, which we seek to explore.

3.1 Datasets

We manually created two NLQ datasets for scientific literature discovery. A NLQ can specify conditions on paper metadata (e.g., publication year) and content (e.g. specific keywords to appear in the paper). We also allow for combining conditions with the boolean operators AND (e.g., papers are about vaccines and COVID-19) and OR (e.g., papers are authored by John Moore or Steven Johnes).

The first **Metadata** dataset contains requirements on paper metadata and content and was constructed as follows:

1. We first selected a research domain (e.g., biomedical science), then came up with relevant keywords (e.g., vaccines) and publisher (e.g., Nature).

2. Next, we continuously added conditions on the paper metadata until the publisher's search engine could find only very few papers relevant to the topic while meeting all conditions.

3. We consolidated the keywords and the conditions into a NLQ, and linked each NLQ to the papers found on the corresponding publisher's website in Step 2 (target papers).

The second **Content** dataset focuses on restricted paper content and was designed by tying the query with the paper content rather than its metadata. Specifically, this dataset contains NLQs that combine keywords that rarely co-occur within single papers:

1. First, we identified suitable keyword combi-

Statistic	Meta.	Cont.
Queries	30	30
Conditions per query	3 - 9	2 - 4
Tokens per query	18 - 66	16 - 22
Target papers per query	1 - 5	N/A
TTR	0.45	0.32
RTTR	13.85	8.98

Table 1: Basic statistics of the **Metadata** dataset. TTR - Type-Token Ratio, RTTR - Root Type-Token Ratio (Torruella and Capsada, 2013).

nations (e.g. "connectomics", "entropy maximization", "diffusion tensor imaging").

2. Then, we translated each combination into an NLQ: "Find scientific articles that include precisely the following set of keywords:...'. We also experimented with increasing the verbosity of queries by further characterizing the keyword combinations, see Appendix D.

3. For this dataset, there are no predefined target papers. Every paper retrieved by the engines is classified as correctly retrieved if it contains all target keywords.

In total, we constructed 30 NLQs for **Metadata** and 30 NLQs for **Content**. Figure 1 shows example NLQs and Table 1 lists basic statistics.

3.2 Literature discovery systems

We compared search engines powered by **lexical** similarity, **semantic** similarity, or **chatbots**. A more detailed description of each platform is available in Appendix E.

Lexicality *Google Scholar*[5] is a free search engine that indexes scholarly works and relies primarily on lexical matching. It retrieves results based on exact keywords and phrases, making search accuracy dependent on precise wording.

Semantics *Semantic Scholar* (Kinney et al., 2023) is one of the largest open-sourced platforms for scientific literature discovery. It uses a two-stage search engine: the first stage efficiently finds many relevant papers and the second stage more carefully reranks these papers by semantic relevance. The open-sourced search engine of *SciLit* (Gu and Hahnloser, 2023) is similar, but additionally supports sophisticated metadata filtering options. The closed-sourced *Elicit*[6] is an LLM-powered platform for biomedical literature using semantic similarity[7].

Chatbots *Consensus*[8] and *Perplexity*[9] are both popular closed-sourced chatbots for getting answers from real-world information sources. We also included *Floatz*[10] and *Zeta-Alpha*[11], two closed-sourced platforms combining LLMs, semantic search and indexing. Additionally, we compared against *ChatGPT-4o*[12], a general-purpose closed-sourced chatbot.

3.3 NLQ Parsing

Additionally, we investigated how parsing the original NLQ into a structured query aligning with the specific engine's specifications affects retrieval performance. This part of the analysis was possible only for platforms which specify their structured query format.

For *SciLit*[13] and *Semantic Scholar*[14], we followed the provided documentation on their discovery engines to convert NLQs into structured queries. Next, we benchmarked the retrieval performance using the structured queries. The conversion was performed with a GPT-4o-mini model under a few-shot setting (Appendix A).

Note that our dataset's queries were not fully compatible with the *Semantic Scholar* API: (1) strict keyword filtering (exact word matches), (2) logical OR functionality for metadata fields (e.g., limiting to papers from *Nature* OR *Science*), and (3) exclusion queries (e.g., ignoring specific authors) are not supported. To optimize our use of *Semantic Scholar*, we curated a small dataset comprising 15 queries that align with the platform's API constraints (Appendix B). For the **Metadata** dataset, content and author requirements were merged into the "query" field, while venue and year constraints were assigned to their respective fields. OR conditions were interpreted as AND conditions, and exclusions were disregarded. For the **Content** dataset, keyword-based constraints were entered into the "query" field.

Google Scholar lacks an official API constructing structured searches. Based on empirical anal-

[5] scholar.google.com
[6] elicit.org
[7] support.elicit.com/en/articles/552705
[8] consensus.app
[9] perplexity.ai
[10] floatz.ai/
[11] zeta-alpha.com/
[12] openai.com/index/hello-gpt-4o/
[13] github.com/nianlonggu/SciLit
[14] api.semanticscholar.org/api-docs

ysis of query structure and results, we proposed a parsing scheme, detailed in Appendix C. However, without an official documentation, optimal query formatting cannot be guaranteed.

3.4 Performance analysis

For both **Metadata** and **Content** queries, we evaluated retrieval performance by comparing each platform's output against the target papers. Papers present in the target list were classified as "correct papers", whereas the non-targets were classified as "incorrect papers". Additionally, any non-existent papers returned by the platforms were categorized as "hallucinated papers". For each platform, we computed average Precision, Recall, and F-1 across the $N = 30$ queries:

$$\text{Precision} = \frac{1}{N}\sum_{q=1}^{N}\frac{\text{Correctly Retrieved}_q}{\text{Total Retrieved}_q}$$

$$\text{Recall} = \frac{1}{N}\sum_{q=1}^{N}\frac{\text{Correctly Retrieved}_q}{\text{Total Targets}_q}$$

$$\text{F1} = \frac{1}{N}\sum_{q=1}^{N} 2 \times \frac{\text{Precision}_q \times \text{Recall}_q}{\text{Precision}_q + \text{Recall}_q}$$

4 Results and discussion

Overall, the examined platforms rarely hallucinated papers. For **Metadata** queries, only Perplexity suggested one hallucinated paper, while all other platforms suggested none. No hallucinations were observed for **Content** queries.

Systems Struggled with Precise Queries Table 2 highlights the challenges most search engines face with precise NLQs. For **Metadata**, precision remains low, except for *SciLit* + LLM parsing, *Google Scholar* + LLM parsing, *ChatGPT-4o*, and *Perplexity*. The highest Recall scores were achieved by *Google Scholar* + LLM parsing, *SciLit* + LLM parsing, *Perplexity*, and *ChatGPT-4o*. In terms of F1 score, *Google Scholar* + LLM parsing, *SciLit* + LLM parsing, *ChatGPT-4o*, and *Perplexity* outperformed other platforms, with *Google Scholar* + LLM parsing delivering the best overall performance. For the **Content** dataset, precision is notably high for *Google Scholar* + LLM parsing, *Google Scholar*, and *Zeta-Alpha*, but remains low for other platforms. Only *Google Scholar* + LLM parsing demonstrated consistently high Recall and F1 scores, making it the top performer on this dataset. *SciLit* and *Semantic Scholar* perform poorly on **Content**, even with LLM parsing.

	Metadata			**Content**		
	P	R	F1	P	R	F1
Elicit	0.08	0.41	0.12	0.03	0.10	0.04
Zeta-Alpha	0.23	0.19	0.20	0.60	0.31	0.39
Consensus	0.08	0.33	0.11	0.01	0.05	0.02
Floatz	0.10	0.10	0.10	0.22	0.09	0.11
Perplexity	0.55	0.57	0.52	0.10	0.09	0.08
ChatGPT-4o	0.61	0.53	0.55	0.19	0.11	0.13
SciLit	0.01	0.01	0.01	0.00	0.03	0.01
+ LLM parsing	0.78	0.76	0.76	0.42	0.17	0.23
Semantic Scholar	0.00	0.00	0.00	0.00	0.00	0.00
+ LLM parsing	0.28	0.48	0.32	0.04	0.05	0.03
Google Scholar	0.03	0.02	0.02	0.64	0.37	0.44
+ LLM parsing	**0.80**	**0.80**	**0.79**	**0.96**	**0.98**	**0.96**

Table 2: Performance of search engines based on **Metadata** and **Content** datasets in terms of **P**recision, **R**ecall, and **F1**. Metrics exclude hallucinated papers. "+ LLM parsing" indicates NLQ converted into a structured query.

The performance metrics are computed by pooling together the target papers obtained across all platforms, but these two platforms use only S2AG (Wade, 2022), which is likely much smaller than the *Google Scholar* paper database. Thus, there were likely many papers that could have never been retrieved in the first place from these two platforms.

LLM Parsing Enhances Discovery Performance For the three platforms that support using structured queries (namely, *SciLit*, *Semantic Scholar*, *Google Scholar*), we found improved retrieval performance on both datasets when we parsed NLQs with an LLM into a structured format. A smaller performance boost was observed for *Semantic Scholar*, likely due to poor compatibility of our dataset's NLQs with its API. The poor performance achieved without LLM parsing is likely due to these three platforms being designed for queries that are keyword- or semantically dense, not for queries that are phrased as instructions.

5 Conclusion

Our findings highlight key strengths and limitations of LLM-based literature discovery systems. While these systems struggle with precise NLQs, LLM parsing significantly enhances retrieval performance, particularly when integrated with structured search engines. This suggests that hybrid approaches combining LLM-based and structured retrieval methods are more promising for literature discovery and could bridge the gap between the flexibility of human-like queries and the structured nature of conventional search engines, effectively

mitigating the challenges posed by ambiguous or instruction-based NLQs. Additionally, LLM-based systems prove valuable in scenarios where structured queries are not feasible or when queries do not conform to strict database formats. Future work should explore refining hybrid methodologies to further optimize retrieval accuracy and relevance.

Limitations

Due to resource constraints, only the free software versions were evaluated. Additionally, we designed only 60 queries because of the extensive work of manually constructing them and of examining the retrieved papers. Also, our precise queries may not be representative of the imprecise queries researchers might submit in practice. In future work, it may therefore be worthwhile to design queries that creatively combine requests for precise content in the midst of imprecise interests, which would call for human evaluation due to the lack of a gold standard.

References

Anirudh Ajith, Mengzhou Xia, Alexis Chevalier, Tanya Goyal, Danqi Chen, and Tianyu Gao. 2024. LitSearch: A retrieval benchmark for scientific literature search. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15068–15083, Miami, Florida, USA. Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Mohammad Dehghan, Mohammad Ali Alomrani, Sunyam Bagga, David Alfonso-Hermelo, Khalil Bibi, Abbas Ghaddar, Yingxue Zhang, Xiaoguang Li, Jianye Hao, Qun Liu, Jimmy Lin, Boxing Chen, Prasanna Parthasarathi, Mahdi Biparva, and Mehdi Rezagholizadeh. 2024. Ewek-qa: Enhanced web and efficient knowledge graph retrieval for citation-based question answering systems. *Preprint*, arXiv:2406.10393.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. *Preprint*, arXiv:2305.14627.

Nianlong Gu, Yingqiang Gao, and Richard H. R. Hahnloser. 2022. Local citation recommendation with hierarchical-attention text encoder and scibert-based reranking. *Preprint*, arXiv:2112.01206.

Nianlong Gu and Richard H.R. Hahnloser. 2023. SciLit: A platform for joint scientific literature discovery, summarization and citation generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 235–246, Toronto, Canada. Association for Computational Linguistics.

Chanwoo Jeong, Sion Jang, Hyuna Shin, Eunjeong Park, and Sungchul Choi. 2019. A context-aware citation recommendation model with bert and graph convolutional networks. *Preprint*, arXiv:1903.06464.

Hemanth Kandula, Damianos Karakos, Haoling Qiu, Benjamin Rozonoyer, Ian Soboroff, Lee Tarlin, and Bonan Min. 2024. Querybuilder: Human-in-the-loop query development for information retrieval. *Preprint*, arXiv:2409.04667.

Binh Thanh Kieu, Inigo Jauregi Unanue, Son Bao Pham, Hieu Xuan Phan, and Massimo Piccardi. 2020. Learning neural textual representations for citation recommendation. *Preprint*, arXiv:2007.04070.

Rodney 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 Graham, Fangzhou Hu, Regan Huff, Daniel King, Sebastian Kohlmeier, Bailey Kuehl, Michael Langan, Daniel Lin, Haokun Liu, Kyle Lo, Jaron Lochner, Kelsey MacMillan, Tyler Murray, Chris Newell, Smita Rao, Shaurya Rohatgi, Paul Sayre, Zejiang Shen, Amanpreet Singh, Luca Soldaini, Shivashankar Subramanian, Amber Tanaka, Alex D. Wade, Linda Wagner, Lucy Lu Wang, Chris Wilhelm, Caroline Wu, Jiangjiang Yang, Angele Zamarron, Madeleine Van Zuylen, and Daniel S. Weld. 2023. The semantic scholar open data platform. *Preprint*, arXiv:2301.10140.

Janice Y. Kung. 2023. Elicit. *The Journal of the Canadian Health Libraries Association*, 44(1):15–8. © Kung. No competing interests declared.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *Preprint*, arXiv:2411.07763.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. *Preprint*, arXiv:2304.09848.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Smriti Mallapaty. 2024. Can google scholar survive the ai revolution? *Nature*, 635:797–798.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes. *Preprint*, arXiv:2203.11147.

Mahdi Naser Moghadasi and Yu Zhuang. 2020. Sent2vec: A new sentence embedding representation with sentimental semantic. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 4672–4680.

Chantal Montgomery, Haruna Isah, and Farhana Zulkernine. 2020. Towards a natural language query processing system. *Preprint*, arXiv:2009.12414.

Rodrigo Nogueira, Zhiying Jiang, Kyunghyun Cho, and Jimmy Lin. 2020. Navigation-based candidate expansion and pretrained language models for citation recommendation. *Preprint*, arXiv:2001.08687.

Mimansha Patel and Nitin Patel. 2019. Exploring research methodology: Review article. *International Journal of Research and Review*, 6(3):48–55. Review Article.

Yuriy A. Portnov. 2018. Change in event horizon surface area as the source of nonmetricity field. *International Journal of Geometric Methods in Modern Physics*, 15(06):1850104.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Surovi Saikia and Manobjyoti Bordoloi. 2019. Molecular docking: Challenges, advances and its use in drug discovery perspective. *Current Drug Targets*, 20(5):501–521.

Johannes Schleith, Hella-Franziska Hoffmann, Milda Norkute, and Brian Cechmanek. 2022. Human-in-the-loop information extraction increases efficiency and trust. In *Mensch und Computer 2022 – Workshopband*, Darmstadt, Germany. Gesellschaft für Informatik e.V.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2024. Is chatgpt good at search? investigating large language models as re-ranking agents. *Preprint*, arXiv:2304.09542.

Joan Torruella and Ramon Capsada. 2013. Lexical statistics and tipological structures: A measure of lexical richness. In *5th International Conference on Corpus Linguistics (CILC2013)*, volume 95, pages 447–454. Elsevier Ltd.

Alex D. Wade. 2022. The semantic scholar academic graph (s2ag). In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 739, New York, NY, USA. Association for Computing Machinery.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Seq2SQL: Generating structured queries from natural language using reinforcement learning.

A LLM parsing into structured queries (SciLit)

To parse the NLQs from the **Metadata** dataset into the structured format required by the SciLit API, we used GPT-4o with the prompt in Figure 2.

Example input query: "Show me publications by Yuriy Portnov in the International Journal of Geometric Methods in Modern Physics from 2010 to 2020 but not 2013, related to black holes or dark matter, and especially articles that discuss: The event horizon's impact on surrounding matter. "

Output structured query: {"Semantic Query": "The event horizon's impact on surrounding matter.", "Keywords": ["Author.FullName: Yuriy Portnov", "Venue: International Journal of Geometric Methods in Modern Physics", "2010..2020", "!2013", "black holes|dark matter"]}.

The prompt for **Content** queries is in Figure 3.

Example input query: "Find scientific articles that include precisely the following set of keywords: 'Piezoelectric materials', 'cellular mechanotransduction', 'ultrasound stimulation'. "

Example output (structured query): ['Piezoelectric materials', 'cellular mechanotransduction', 'ultrasound stimulation']

B Semantic Scholar Custom Dataset and LLM parsing into structured queries (Semantic Scholar)

As the queries in **Metadata** and **Content** are not fully suitable for the *Semantic Scholar* API, we composed a smaller dataset of 15 queries that are fully suitable with their API, **Semantic Scholar Custom**. Queries in **Semantic Scholar Custom** are much simpler than in **Metadata** and **Content**. Specifically, they do not include strict keyword filtering (returning papers containing an exact word match), OR functions (e.g., papers published in Nature OR Science), or exclusions (e.g., papers not authored by Steven Jones, or papers published between 2010 and 2020 but excluding 2018).

Example query from Semantic Scholar Custom dataset: "Find papers on deep reinforce-

I will give you a query text, your task is to extract two sources of information from the text: 1) Semantic Query and 2) Keywords. This query text is a natural language about how I want to query the scientific literature database. You should parse the query text and extract the information in the following steps:

Step 1: Identify the semantic query. You need to inspect the query text and check if there is any text (e.g., some sentences or paragraphs) that the user intends to use as the semantic query to find semantically similar papers. If there is no semantic query specified in the query text, then set the semantic query as an empty string "".

Step 2: Extract keywords. You need to parse the query text and extract keywords mentioned in the query text that are supposed to be used as filters when doing search. The keywords include four and ONLY 4 types:
1. AuthorFullNames: After extracting all authors' full names, prefix each extracted author name with a special string "Author.FullName:".
2. Venue: Extract venue or journal mentioned in the query text, and prefix each extracted venue with a special string "Venue:".
3. PublicationDate: Extract keywords of years or a range of years. If the publication date keywords are a range of years, express the year keywords in the form "Start-Year..End-Year". For individual years, extract the year itself.
4. GeneralKeywords: Extract the keywords that are mentioned in the query text but do not belong to other keyword types. Extract the keywords as they are (maximum three words). Do not copy the semantic query directly as a general keyword, and correct any spelling mistakes in the extracted keywords.

Step 3: Check the NOT logic operation for each extracted keyword. Prefix excluded keywords with "!" where appropriate.

Step 4: Check the OR logic operation between multiple post-processed keywords in Step 3. If there is an OR logic specified between keywords, use the "|" character to join them.

Step 5: Convert the extracted semantic query and the post-processed and extracted keywords into a machine-readable JSON format:
==== Start of the JSON ====
{
"Semantic Query": Put the extracted semantic query here,
"Keywords": Put the post-processed and extracted keywords as a list [k1, ..., kn]
}
==== End of the JSON ====

Have a look at a few examples below:

Example 1:
Query: Find the papers of Jimmy White and Tom Anderson from 2010 to 2020 but not in 2015, published in Nature or Science, on the topic of neuron morphology or machine lerning but not animal behavior, especially related to the statement like: axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits.
Keywords: { "Semantic Query": "Axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits.", "Keywords": ["Author.FullName:Jimmy White", "Author.FullName:Tom Anderson", "Venue:Nature|Venue:Science", "2010..2020", "!2015", "neuron morphology|machine learning", "!animal behavior"] }

Example 2:
Query: Show me papers of John Wick or Robert Smith, about zebra finch but not zebra fish, published on nature communications or PLOS Biology from 2010 to 2024 but not the year 2018. Especially show me the papers related to the content: Juvenile birds learn from adults.
Keywords: { "Semantic Query": "Juvenile birds learn from adults.", "Keywords": ["Author.FullName:John Wick|Author.FullName:Robert Smith", "Venue:Nature Communications|Venue:PLOS Biology", "2010..2024", "!2018", "zebra finch", "!zebra fish"] }

Example 3:
Query: I want to search for papers related to machine learning and zebra finch, authored by Anja Zai and, from 2020 to 2020.
Keywords: { "Semantic Query": "", "Keywords": ["Author.FullName:Anja Zai", "2020..2022", "machine learning","zebra finch"] }
In Example 3, the query contained no text that can be attributed to semantic query, therefore I set the semantic query as an empty string.

Following the instruction above, please parse the following query text step by step:

Figure 2: Prompt for GPT-4o to parse **Metadata** queries into a structure suitable for SciLit.

I will give you a query text, and your task is to extract a list of keywords that should appear in the retrieved papers according to the query. Have a look at the few examples below:

Example 1:
Query: "Find scientific articles that that include precisely the following set of keywords: 'mechanotransduction', 'photosynthesis', 'Calvin Cycle'. "
Keywords: ['mechanotransduction', 'photosynthesis', 'Calvin Cycle']

Example 2:
Query: "Find scientific articles that that include precisely the following set of keywords: 'red blood cells', 'glucometer', 'diabetes'. "
Keywords: ['red blood cells', 'glucometer', 'diabetes']

Following the instruction above, please parse the following query text step by step:

Figure 3: Prompt for GPT-4o to parse **Content** queries into a structure suitable for SciLit.

ment learning authored by David Silver published in NeurIPS since 2021."

We then benchmarked the performance of Semantic Scholar on **Semantic Scholar Custom** queries without LLM parsing and with LLM parsing in two different versions. Version v1, the structured query had 'query', 'venue', 'year', and 'author' parameters, while version v2 had only 'query', 'venue', and 'year' parameters, while the information corresponding to the author was included in the 'query' parameter.

For version **v1**, we used the prompt in Figure 4.

Example input query: "Find papers on deep reinforcement learning authored by David Silver published in NeurIPS after 2021."

Example output (structured query): {'query': 'deep reinforcement learning', 'venue': 'NeurIPS', 'year': '2021-', 'author': 'David Silver'}

For version **v2**, we used the prompt in Figure 5.

Example input query: "Find papers on deep reinforcement learning authored by David Silver published in NeurIPS since 2021."

Example output (structured query): 'query': 'deep reinforcement learning David Silver', 'venue': 'NeurIPS', 'year': '2021-'

As shown in Table 3, LLM parsing v2 yielded the best performance. Hence, we used this version of parsing for the **Metadata** and **Content** datasets.

	P	R	F1
Semantic Scholar	0.00	0.00	0.00
+ LLM parsing_v1	0.01	0.02	0.01
+ LLM parsing_v2	**0.07**	**0.09**	**0.07**

Table 3: Comparison of Semantic Scholar performance based on benchmark queries from the Semantic Scholar custom dataset. Metrics include **P**recision, **R**ecall, and **F1**.

C LLM parsing into structured queries (Google Scholar)

We parsed the NLQs from **Metadata** into the structure that we speculate is consistent with Google Scholar using GPT-4o with the prompt in Figure 6.

Example input query: "I need papers written by Daniel Sheridan and Probir Chakravarty published after 1999 about pituitary stem cells and their roles in regulation of reproductive functions. "

Example output (structured query): (author:"Daniel Sheridan" AND author:"Probir Chakravarty") "pituitary stem cells" "regulation of reproductive functions" year(2000:)

Then, we transformed each structured query into a URL for Google Scholar by extracting relevant filter parameters and encoding them properly. For the above example, the resulting URL is as follows:

```
https://scholar.google.com/scholar?hl=en&
as_sdt=0%2C5&as_q=%28author%3A%22Daniel%
20Sheridan%22%20AND%20author%3A%22Probir%
20Chakravarty%22%29%20%20%22pituitary%20stem%
20cells%22%20%22regulation%20of%20reproductive%
20functions%22&as_epq=&as_oq=&as_occt=any&as_
sauthors=&as_publication=&as_ylo=2000
```

For **Content**, we followed Appendix A.

D Content Dataset - Expanded Query

For each keyword combination in the **Content** dataset, an expanded query was constructed as follows: "Give me the papers in {domain} on the topic of {topic} that contain precisely the following keywords: {keywords}". Domain and Topic were manually specified for each query given the key-

I will give you a text that describes how I want to query the scientific literature database. You task is to parse and extract (1) the semantic query, and (2) the filter parameters from the text. Follow these steps:

Step 1: Extract filter parameters that should be used as filters on the papers to be returned. If a filter parameter is described as a term to be avoided, do not extract it. For each of the three filter parameters types below, extract the corresponding information and process it as follows:
1. venue: Identify all publication venues, conferences, or journal names mentioned, then concatenate them with commas but no spaces.
2. year: Identify the first requirement on publication year mentioned. If exactly one year is described, then extract just the year. If a range of years is described, then express the range by in the form "start-end". If only the start year is described, then write it as "start-", and if only the end year is described, then write it as "-end".
3. author: Identify all author names mentioned and concatenate them with commas but no spaces.

Step 2: Identify the semantic query. This refers to any part of the text describes what the papers of interest should be about, and it may be words, phrases, sentences, or paragraphs. It should include all meaningful phrases that were not extracted above. If no semantic query is described in the text, then set the semantic query as an empty string "".

Step 3: Put the semantic query and the processed filter parameters together into a machine-readable JSON object:
==== Start of the JSON ====
{
"query": Put the extracted semantic query here,
"venue": Put the extracted venue requirement here,
"year": Put the extracted year requirement here,
"author": Put the extracted author requirement here
}
==== End of the JSON ====

Here are a few examples:

Example 1:
Query: Find the papers of Jimmy White and Tom Anderson published in Nature or Science in 2010, on the topic of neuron morphology or machine learning but not animal behavior, especially related to the statement like: axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits.
Output: {
"query": "Axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits neuron morphology machine learning",
"year": "2010",
"venue": "Nature,Science",
"author": "Jimmy White,Tom Anderson
}

Example 2:
Query: Show me papers of John Wick and Robert Smith, about zebra finch but not zebra fish, published in nature communications or PLOS Biology before 2020. Especially show me the papers related to the content: Juvenile birds learn from adults.
Output: {
"query": "Juvenile birds learn from adults zebra finch",
"venue": "Nature Communications,PLOS Biology"
"year": "-2019",
"author": "John Wick,Robert Smith"
}

Example 3:
Query: I want to search for papers related to machine learning and biomedical applications, authored by Yoshua Bengio since 2023.
Output: {
"query": "machine learning and biomedical applications",
"venue": "",
"year": "2023-",
"author": "Yoshua Bengio"
}
Following the instructions above, please parse the following query description text:

Figure 4: Prompt for GPT-4o to parse **Custom** queries into a structure suitable for Semantic Scholar, version 1.

I will give you a text that describes how I want to query the scientific literature database. You task is to parse and extract (1) the semantic query, and (2) the keywords from the text. Follow these steps:

Step 1: Extract keywords that should be used as filters on the papers to be returned. If a keyword is described as a term to be avoided, do not extract it. For each of the three keyword types below, extract the corresponding information and process it as follows:
1. venue: Identify all publication venues, conferences, or journal names mentioned, then concatenate them with commas but no spaces.
2. year: Identify the first requirement on publication year mentioned. If exactly one year is described, then extract just the year. If a range of years is described, then express the range by in the form "start-end". If only the start year is described, then write it as "start-", and if only the end year is described, then write it as "-end".

Step 2: Identify the semantic query. This refers to any part of the text describes what the papers of interest should be about, and it may be words, phrases, sentences, or paragraphs. It should include all meaningful phrases that were not extracted above. If no semantic query is described in the text, then set the semantic query as an empty string "".

Step 3: Put the semantic query and the processed keywords together into a machine-readable JSON object:
==== Start of the JSON ====
{
"query": Put the extracted semantic query here.
"venue": Put the extracted venue requirement here.
"year": Put the extracted year requirement here.
}
==== End of the JSON ====

Here are a few examples:

Example 1:
Query: Find the papers of Jimmy White and Tom Anderson published in Nature or Science in 2010, on the topic of neuron morphology or machine learning but not animal behavior, especially related to the statement like: axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits.
Output: {
"query": "Axonal and dendritic arbors as key functional components of neural processing and fundamental determinants of neural circuits Jimmy White Tom Anderson neuron morphology machine learning",
"year": "2010",
"venue": "Nature,Science",
}

Example 2:
Query: Show me papers of John Wick or Robert Smith, about zebra finch but not zebra fish, published in nature communications or PLOS Biology before 2020. Especially show me the papers related to the content: Juvenile birds learn from adults.
Output: {
"query": "Juvenile birds learn from adults John Wick Robert Smith zebra finch",
"venue": "Nature Communications,PLOS Biology"
"year": "-2019",
}

Example 3:
Query: I want to search for papers related to machine learning and biomedical applications, authored by Yoshua Bengio since 2023.
Output: {
"query": "machine learning and biomedical applications Yoshua Bengio",
"venue": "",
"year": "2023-"
}

Following the instructions above, please parse the following query description text step by step:

Figure 5: Prompt for GPT-4o to parse **Custom** queries into a structure suitable for Semantic Scholar, version 2.

Help me translate natural language queries into structured format required by Google Scholar. Follow these steps:

Step 1: First, extract author information from the query (if there is any). Use author's name, and prepend it with 'author:'. For example: 'author:Jones'.
If the query specifies two authors with AND operation, specify that as follows: (author:"Gao" AND author:"Gu").
If the query specifies two authors with OR operation, specify that as follows: (author:"Gupta" OR author:"Srivastava").

Step 2: Then, extract venue information from the query (if there is any).
Prepend it with 'source:'. For example: 'source:Nature'. If the query specifies two venues with OR operation, specify that as follows: (source:"Science" OR source:"Nature").

Step 3: Then, extract year information from the query (if there is any).
If the query specifies a years range, specify that as follows: year(2020:2022). If the query specifies a given year, specify that as follows: year(2015).
If the query specifies upper (inclusive) bound for year, specify that as follows: year(:2000).
If the query specifies lower (inclusive) bound for year, specify that as follows: year(2015:).

Step 4: Then, extract keywords from the query (if there are any). Put them inside double quotation mark, for example "quantum mechanics".
If there is an OR operation, specify this as follows: ("deep learning" OR "reinforcement learning").
If there is AND operation, specify it as follows: "diabetes" "glucometer".

Step 5: In addition, you can include NOT operation using a dash in the following way: -"dataset" (when we want to exclude papers containing the keywords "dataset").
You can also use the NOT operation for year: -year:2009 (excludes papers published in 2009).
You can use the NOT operation for author: (author:"Johns" -author:"Smiths") (when you want to retrieve papers written by Johns but excluding the ones co-authored by Smiths).
You can use the NOT operation for venue: -source:"arXiv" (when you want to exclude papers published in arXiv).

Step 6: Finally, concatenate all conditions into one structured queries using a single space to separate different parameters.

Have a look at the few examples below:

Example 1:
Query: Find papers authored by Saleska and Mackelprang between 2013 and 2020, particularly the ones mentioning permafrost and Arctic environments.
Target output: (author:"Saleska" AND author:"Mackelprang") year(2013:2020) "permafrost" "Arctic environments"

Example 2:
Query: Find papers published in Science or Nature about deep learning or reinforcement learning.
Target output: (source:"Science" OR source:"Nature") ("deep learning" OR "reinforcement learning")

Example 3:
Query: Give me papers written by Yuriy Portnov, published in International Journal of Geometric Methods in Modern Physics, about black holes or dark matter.
Target output: author:"Yuriy Portnov" source:"International Journal of Geometric Methods in Modern Physics" ("black holes" OR "dark matter")

Following the instructions above, please parse the following query description text step by step:

Figure 6: Prompt for GPT-4o to parse **Metadata** queries into a structure suitable for Google Scholar.

word combination. Example: "Give me research papers in Biotechnology on the topic of Enzyme Engineering that contain precisely the following keywords: 'Metalloenzyme catalysis', 'directed evolution', 'biofuel production'. "

The retrieval results obtained with the expanded query are summarized in Table 4.

Method	P	R	F1
Google Scholar	**0.56**	**0.30**	**0.37**
Elicit	0.05	0.10	0.06
Zeta-Alpha	0.52	0.28	0.34
Consensus	0.02	0.08	0.03
Perplexity	0.12	0.11	0.10
Floatz	0.04	0.02	0.03
GPT-4o	0.17	0.15	0.14
Semantic Scholar	0.00	0.00	0.00
SciLit	0.00	0.00	0.00

Table 4: Performance of search engines based on **Content** dataset with expanded queries in terms of **P**recision, **R**ecall, and **F1**. Metrics exclude hallucinated papers.

E Search Engines

The majority of the platforms included in our analysis are not open-source and do not fully disclose the details of their underlying databases or search algorithms. As a result, our descriptions are based on publicly available information and general observations about their functionality rather than a complete technical breakdown of their inner workings.

- **Google Scholar** is a widely used academic search engine that indexes scholarly literature from a vast array of sources, including publisher websites, institutional repositories, preprint servers, and open-access archives. While its exact database composition is proprietary, it continuously crawls and aggregates research papers, theses, books, and conference proceedings. For search, it primarily performs a keyword-based search, supporting Boolean operators, phrase searches, and field-specific queries. Beyond simple keywords matching, Google Scholar incorporates citation-based ranking to prioritize influential papers. Additionally, it employs semantic search techniques to understand the query intent and retrieve conceptually relevant papers (Mallapaty, 2024).

- **Semantic Scholar** primarily uses the Semantic Scholar Open Research Corpus (S2ORC) (Lo et al., 2020) as its database. Unlike traditional keyword-based search engines, it leverages Machine Learning to enhance search relevance and understanding. When a user submits a query, Semantic Scholar applies keyword-based search, citation analysis, and semantic search techniques to retrieve the most relevant papers. It ranks the results based on factors like citation count, influence score, and content similarity, rather than just exact keyword matches.

- **Consensus** leverages the same database as Semantic Scholar, updating it on a monthly basis. For paper searching, it integrates LLMs with a specialized Vector Search system. When a user enters a textual query into the chatbot interface, the input undergoes preprocessing (e.g., stopword removal). Subsequently, a hybrid approach combining keyword search and Vector Search is applied to the abstracts and titles of all papers in the database to determine a relevance score for each document. This score is then refined using additional metadata, such as citation count, study design, and publication date, to re-rank the results and generate a final list of the top 10 most relevant papers.

- **Perplexity** uses LLMs like GPT-4o[15] and Claude 3[16] to interpret the context and nuances of user queries. After a user enters a query, the query is first passed through an LLM. Then, a real-time web search is conducted, retrieving information from sources such as as articles, websites, and academic journals. The extracted insights are then synthesised into a response. Following, citations to sources are added to the output text, enabling users to verify information.

- **SciLit** is the only fully open-source platform in our analysis. It utilizes multiple scientific text corpora (S2ORC (Lo et al., 2020), PMCOA[17], arXiv[18], bioArxiv[19], and medRxiv[20]), structuring each corpus as a separate SQLite database. It indexes research papers using

[15] https://openai.com/index/hello-gpt-4o/
[16] https://claude.ai/
[17] https://healthdata.gov/dataset/PubMed-Central-Open-Access-Subset-PMC-OA-/3vwy-a2x4/about_data
[18] https://info.arxiv.org/help/bulk_data.html
[19] https://www.biorxiv.org/tdm
[20] https://www.medrxiv.org/

both an inverted index for keyword-based retrieval and an embedding index for semantic search. SciLit first applies Boolean filtering to refine results based on keywords, and then ranks the filtered papers by computing cosine similarity between their Sent2Vec (Moghadasi and Zhuang, 2020) embeddings and the user query. Finally, SciBERT (Beltagy et al., 2019) is used for re-ranking, ensuring that the most relevant papers appear at the top.

- **Elicit** utilizes the Semantic Scholar database, updating the collection weekly with newly added research papers. Unlike traditional search engines, it does not rely on keyword-based queries or controlled vocabulary. Instead, users are encouraged to input full research questions, such as "How does iron supplementation affect anemia?". Upon receiving a query, Elicit retrieves the eight most relevant papers and extracts key insights or variables based on user preferences (Kung, 2023).

- **Floatz** integrates with a wide range of open-source databases and publisher sources, including Elsevier[21], Clarivate[22], PubMed[23], and preprint repositories. If a specific paper is not available in its databases, it leverages integrations like OpenAlex[24] to retrieve the necessary information. While details about its search functionality are limited, Floatz combines LLMs, semantic search, indexing, and knowledge-building algorithms to process user queries effectively.

- **Zeta-Alpha** operates on its own indexed database, incorporating sources such as arXiv, conference proceedings, blogs, and GitHub repositories. Additionally, users can upload their own documents and references, which are then indexed using the platform's neural search technology. For search, Zeta-Alpha employs a hybrid approach that combines traditional keyword-based search - supporting Boolean operators, phrase searches, and field-specific queries - with neural vector search and fine-tuned LLMs, such as Zeta-Alpha-E5-Mistral[25].

- **GPT-4o** Unlike other platforms in our analysis, GPT-4o is a general-purpose AI model designed for a wide range of tasks, not specifically for retrieving scientific literature. It does not index or query academic databases like Google Scholar or Semantic Scholar but instead generates responses based on pre-trained knowledge.

F Implementation details

We used the free versions of all listed platforms, entering each query manually into their search interfaces and recording the retrieved papers. For chatbot-based search systems, paper titles were manually extracted from the responses. If a chatbot found no exact matches but suggested alternatives, we labeled it as "no paper retrieved." No platform-specific filters were applied to ensure evaluation was based solely on NLQs.

For search engines like *Consensus* and *Elicit*, which returned many papers, we analyzed only papers appearing before pressing the "Load More" or "More Results" buttons, which is 10 papers at most. Documents other than research papers, such as books, reviews, and theses, were excluded.

[21] https://www.elsevier.com/
[22] https://mjl.clarivate.com/home
[23] https://pubmed.ncbi.nlm.nih.gov/
[24] https://openalex.org/
[25] https://huggingface.co/zeta-alpha-ai/Zeta-Alpha-E5-Mistral