THE UNIVERSITY OF CHICAGO

TOWARD A COMPUTABLE SCIENTIFIC CORPUS: RETRIEVAL-AUGMENTED REASONING SYSTEMS FOR SCIENTIFIC DISCOVERY ON EXASCALE SUPERCOMPUTERS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY
OZAN GÖKDEMIR

CHICAGO, ILLINOIS
JUNE 2026

ORCID iD: https://orcid.org/0000-0001-5299-1983

This thesis is dedicated to the memory of the late Mr. Akgün Temizer and the late Mr. Erol Üçer — two giants whose belief in students they would never meet made journeys like mine possible.

Hayatta en hakiki mürşit ilimdir.

The truest guide in life is science.

— Mustafa Kemal Atatürk

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Achieving this degree is an honor and privilege that is the culmination of almost two decades of ceaseless effort. It has been a long, arduous, but rewarding journey — one that I could not have completed without the support and guidance of many.

I would like to thank my advisor Prof. Rick L. Stevens, and my committee members Prof. Ian T. Foster and Dr. Arvind Ramanathan for their insightful advice, patience, and understanding. Being their student and apprentice-in-science has been a privilege I will cherish for the rest of my life.

I also would like to express sincere appreciation for my colleagues Peng Ding, Alex Brace, Kyle Hippe, Carlo Siebenschuh, Azton Wells, Brian Hsu, Priyanka Setty, and countless others. Thank you for all our stimulating debates and discussions; and thank you for sharing the burden and excitement of navigating the ambiguity of the frontier of human knowledge.

I would like to offer my wholehearted gratitude for my dear parents Ahmet and Birgül for the unwavering trust they placed in me throughout my life. I firmly believe that humanity would greatly benefit from a dissertation from you two on how to raise children. I offer this accomplishment as a humble attempt to pay you back for your selfless sacrifice.

I also want to thank my dearest little sisters Hazal and Ada for giving me a great purpose to succeed in life — so that I can inspire you today, and support you whenever you are in need of your big brother.

Finally, I would like to thank my dearest Oguljan for supporting me with patience and understanding as I prepared to defend this dissertation— I know it has not always been easy to be around so much stress, yet, you were there for me to lean on when I needed you the most.

ABSTRACT

The rate of growth in scientific output has surged past the human cognitive throughput of information ingestion. As submissions to biomedical publication databases like PubMed alone exceed over three articles *per minute*, researchers are consistently surveyed to read 22 articles per month on average. Large Language Models (LLMs) emerged as powerful engines for ingesting and synthesizing this growing corpus. However, their innately probabilistic working principle is reliant on their static and obsolescent pretraining data. Moreover, training corpora of frontier LLMs are compiled for breadth and broad applicability, not for scientific depth and domain expertise. Naturally, this composition renders LLMs unreliable and prone to hallucination in knowledge-intensive scientific tasks. Retrieval-Augmented Generation (RAG) addresses these shortcomings by equipping LLMs with an extrinsic knowledge source at inference time, allowing for ingestion of additional knowledge without altering the model parameters. Yet scaling RAG to handle the deluge of scientific output– in the order of millions of documents– introduces challenges at every stage of the pipeline. To name a few, parsing dense multi-modal PDFs, encoding domain-specific, terminology-rich text, evaluation models on unmistakably contaminated benchmarks, and orchestrating hundreds of compute nodes with thousands of GPUs to simply overcome the sheer scale of the problem.

This dissertation presents the design, implementation, and evaluation of retrieval-augmented reasoning systems that operate at the scale of millions of scientific documents and leverage exascale supercomputing infrastructure to transform how scientists– human and AI alike– interface with the growing body of scientific literature.

We base this dissertation on three principal contributions. First, we present **HiPerRAG**, a distributed high-performance computing workflow for RAG over scientific literature that indexes over 3.6 million full-text articles using three leadership-class supercomputers. Through the Oreo multimodal parser and the ColTrast query-aware encoder fine-tuning objective, HiPerRAG empowers open-source models with retrieval and leads them to outperform pro-

prietary frontier LLMs on scientific question-answering tasks. Second, we introduce a scalable **automated multiple-choice question answering (MCQA) benchmark generation pipeline** that produces hundreds of thousands of provenance-tracked questions from tens of thousands of full-text articles. Through this work, we identify *distillation through retrieval* as a viable alternative to weight-based distillation. By treating frontier-model reasoning traces as retrievable artifacts rather than training or fine-tuning signals, we enable small language models such as TinyLlama-1.1B to achieve a 4× improvement in domain accuracy and bring several 7-8B parameter models within striking distance of frontier performance on an expert-annotated radiation oncology examination. Third, we introduce **Swarm Retrieval**, a forward-looking paradigm in which documents are not treated as passive points in a vector space, but as agents embodying documents that can judge their own relevance to a given query in an interpretable manner. This paradigm challenges the status-quo of documents as inert static entries, and reimagines them as active participants of the retrieval process. Taken together, these contributions chart a path toward a *computable scientific corpus* which we define as a unified, queryable interface over the full body of published science that is technically feasible with current exascale infrastructure.

CHAPTER 1

INTRODUCTION

1.1 The Scientific Literature Crisis

At its core, scientific pursuit is a cumulative endeavor in which every advancement builds upon previous contributions. Those who seek to advance science are obliged to do so with a keen awareness of what has been hitherto accomplished by their predecessors. Therefore, keeping abreast of the latest developments in their field is a crucial prerequisite for scientists to advance it through novel and rigorous hypotheses.

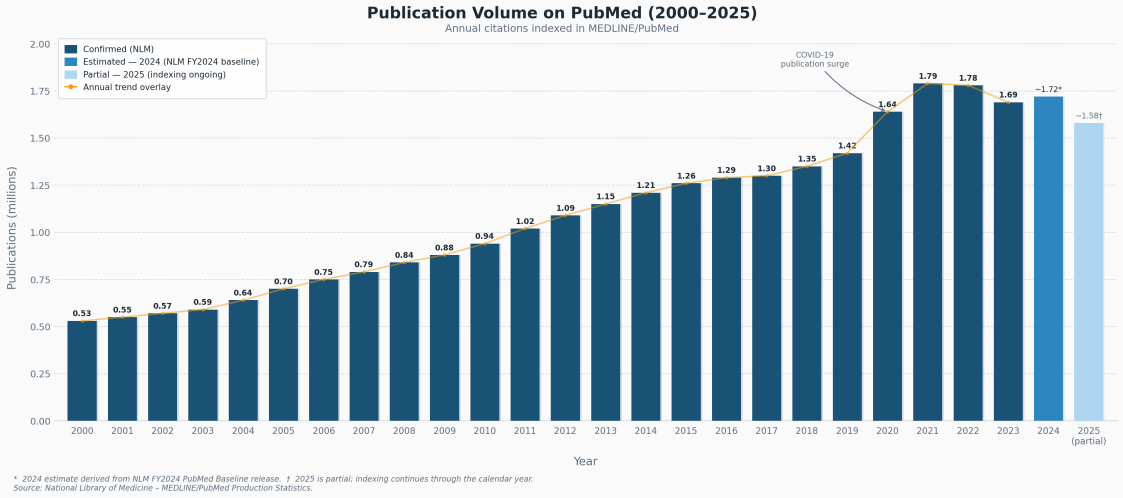

Figure 1.1: Number of articles submitted to the PubMed database annually from 2000 to 2023. Data comes from PubMed which is maintained by the National Center of Biotechnology Information (NCBI) at the National Institutes of Health (NIH)

Yet, the rapid rate of growth in scientific output renders it insurmountable for any individual scientist to navigate the deluge of information [85]. The 2024 State of United States Science and Engineering report by the National Science Foundation reveals that the number of science and engineering articles published in open-access journals annually has surged from 19,000 in 2013 to 992,000 in 2022, marking over a 50-fold increase. This trend of rapid growth extends into subscription-based publishing venues, as well. Hanson et al. report that

Scopus, a major citation abstract database, indexed approximately 896,000 more articles in 2022 than in 2016 (2.82 million vs 1.92 million), corresponding to a 5.6% year-on-year growth over that period [59]. In the biomedical domain alone, PubMed [113] processed approximately 1.69 million articles last year, averaging more than three articles per minute. Clearly, the exponential growth in the global publication volume presents a daunting challenge for scientists seeking to ingest incoming information.

As this publication boom unfolds, scientists appear to have reached a ceiling in the number of papers they can read, comprehend, and retain. Across multiple surveys, scientists report reading on the order of 22 academic articles per month on average – a statistic that has remained essentially flat for over a decade. [157]. We interpret this finding not as a lack of motivation on the part of the scientific community, but rather as a hard biological constraint on the individual scientist. Humans are simply bound by a finite amount of time, attention, and cognitive throughput. In the same vein, Meho reports that roughly 90% of papers published in academic journals are never cited and as many as 50% of them are never read by anyone other than their authors, referees, and journal editors [105].

This disparity between the rate of publication growth and an individual scientist's capacity for ingestion contributes to a citation imbalance whereby a significant portion of scientific research remains underutilized. Nielsen and Andersen report that the top 1% of most-cited scientists increased their cumulative citation share from 14% to 21% between 2000 and 2015 [115]. This concentration of attention not only hinders the dissemination of valuable knowledge but also perpetuates a cycle where highly cited scientists receive even more recognition and resources, widening the gap between the most and least cited researchers [129]. The consequences are tangible: promising avenues for future research remain dormant and unexplored, costly and time-consuming efforts are duplicated, and the diversity of scientific discourse is stifled. Chu and Evans provide empirical support for this view, demonstrating that the growth of large scientific fields is associated with a slowdown of canonical progress,

as new work increasingly consolidates around established paradigms rather than exploring novel directions [28].

In sum, the prevailing *publish or perish* paradigm [45, 114] has transformed literature review from an essential scholarly activity into a *big data* management problem. Scientific output grows at a large rate that no individual can any longer match, and in the absence of computational methods to streamline the process, scientists spend disproportionate time on literature review while a plethora of valuable findings still remain underutilized in rapidly accumulating piles. This discussion proceeds with an introduction of Large Language Models: one of the most transformative and rapidly-adopted technological advancements in processing and generating information which are not exempt from innate shortcomings of their own.

1.2 Large Language Models and Their Limitations for Science

The advent of large language models (LLMs) has provided researchers with powerful tools for processing vast scientific corpora. Built on the Transformer architecture [158], LLMs leverage self-attention mechanisms to capture long-range dependencies in text, enabling them to serve as remarkably effective associative memories over their training data. The scaling of these models—from hundreds of millions of parameters in early models like BERT [39] and GPT-2 [127] to hundreds of billions or even trillions in DeepSeek-V3 [35], Llama 4 [106], GLM-5 [177], and Kimi K2 [81]—has yielded emergent capabilities in reasoning, summarization, and knowledge synthesis that were not anticipated from smaller predecessors [166]. In the scientific domain, LLMs have been deployed across an increasingly ambitious range of tasks, from automated literature-grounded medical question answering [144] and protein sequence design [44] to end-to-end autonomous research pipelines capable of ideation, experimentation, and manuscript generation [99, 54]. We survey specific models in this space in Section 2.1

Despite their utility in associative memory and knowledge synthesis, LLMs face several critical shortcomings when applied to science. First, the parametric memory that an LLM

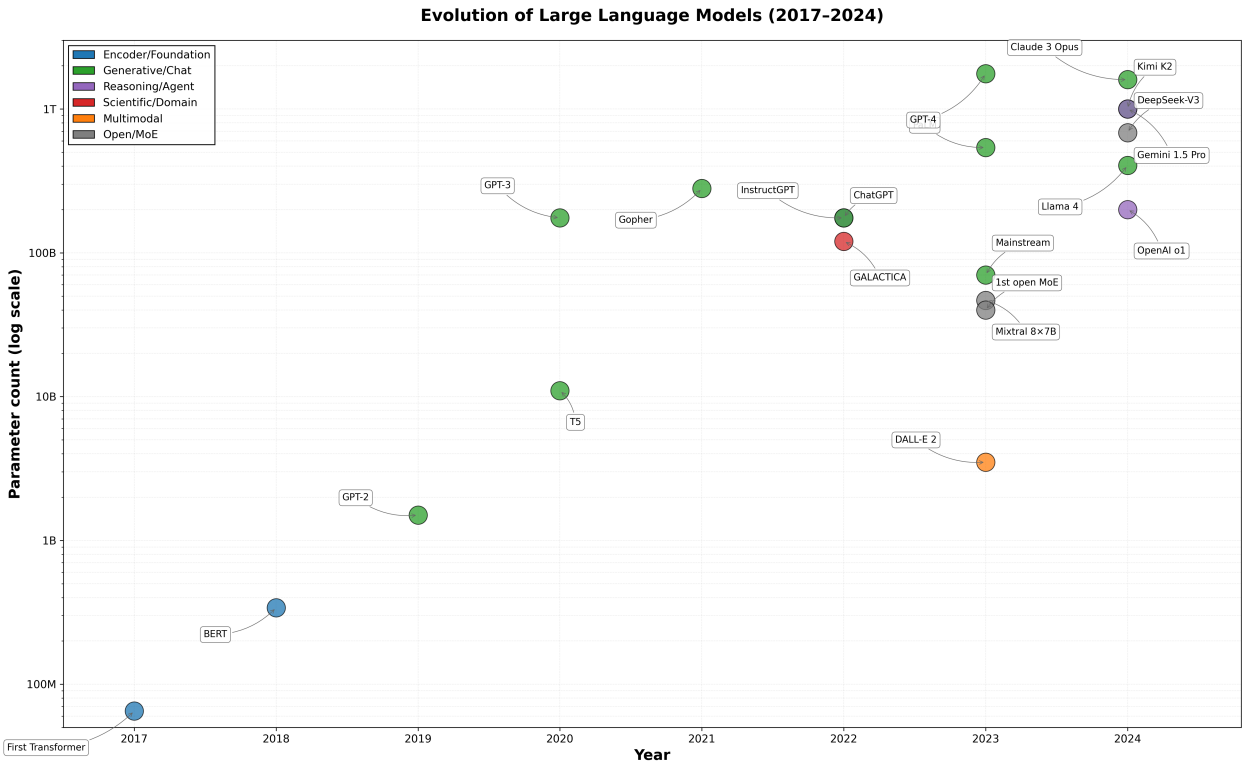

Figure 1.2: Evolution of large language models by parameter count, 2017–2025. Dot size and halo indicate milestone significance. Parameter counts for mixture-of-experts models reflect total parameters; active parameters per token are substantially smaller.

acquires during pretraining is stored in its weights and is static, i.e., by default, base models are unaware of knowledge disseminated after the cutoff date of their pretraining data. Although techniques such as fine-tuning [179] and knowledge injection [172] can remediate this, these are error-prone methods that risk catastrophic forgetting [82, 53] of pre-existing knowledge whereby a model not only fails to learn the new data distribution, but also suffers its prior modeling capabilities to degrade.

Second, general-purpose frontier LLMs are typically trained on web-scale datasets such as The Pile [47], RedPajama [154], and Dolma [148]. These massive datasets favor breadth over depth so as to render commercial frontier models widely useful for as large a user base as possible at the expense of deep specialization in knowledge-intensive endeavors like science [91].

Third, and most critically, LLMs are prone to *hallucinate* plausible yet factually incorrect information – a well-documented limitation [159, 67, 170]. Hallucinations stem from the fact that LLMs model the joint distribution of tokens (finite building blocks of text) observed in their pretraining data. During generation, they sample this distribution in an autoregressive manner whereby the next token is selected simply as the most likely token to follow what the model has hitherto written. Clearly, most probable output is not often the most accurate or up-to-date one. Thus, hallucination remains a daunting challenge in adopting LLMs in scientific contexts where factual grounding is paramount. As an early attempt at a purpose-built scientific *foundation model*, GALACTICA demonstrated both the promise and the peril of large-scale scientific text generation: while capable of impressive knowledge synthesis even in notation-heavy life sciences, the model was retracted from public access within days of its release due to the widely-reported severe hallucinations observed in its scientific claims [150]. Subsequent work by Min et al. on FactScore [107] established that even the state-of-the-art LLMs produce factually inconsistent statements when evaluated at the atomic claim level, reinforcing the conclusion that the most probable token sequence is not always the true one.

1.3 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) was proposed by Lewis et al. [87] to address the aforementioned shortcomings by equipping LLMs with an external knowledge source. Rather than relying solely on parametric memory, RAG integrates neural information retrieval with LLM-based content generation. This approach leverages the ability of LLMs to represent any chunk of text, e.g., a sentence, a paragraph, or an entire document, as a fixed-size embedding vector that encodes semantic relationships akin to meaning. The distance between those vectors, measured by cosine similarity, Euclidian distance, Frobenius distance, or even a learned metric [180, 16] quantifies their relevance and enables retrieval systems to efficiently identify documents that are most pertinent to a given query [79].

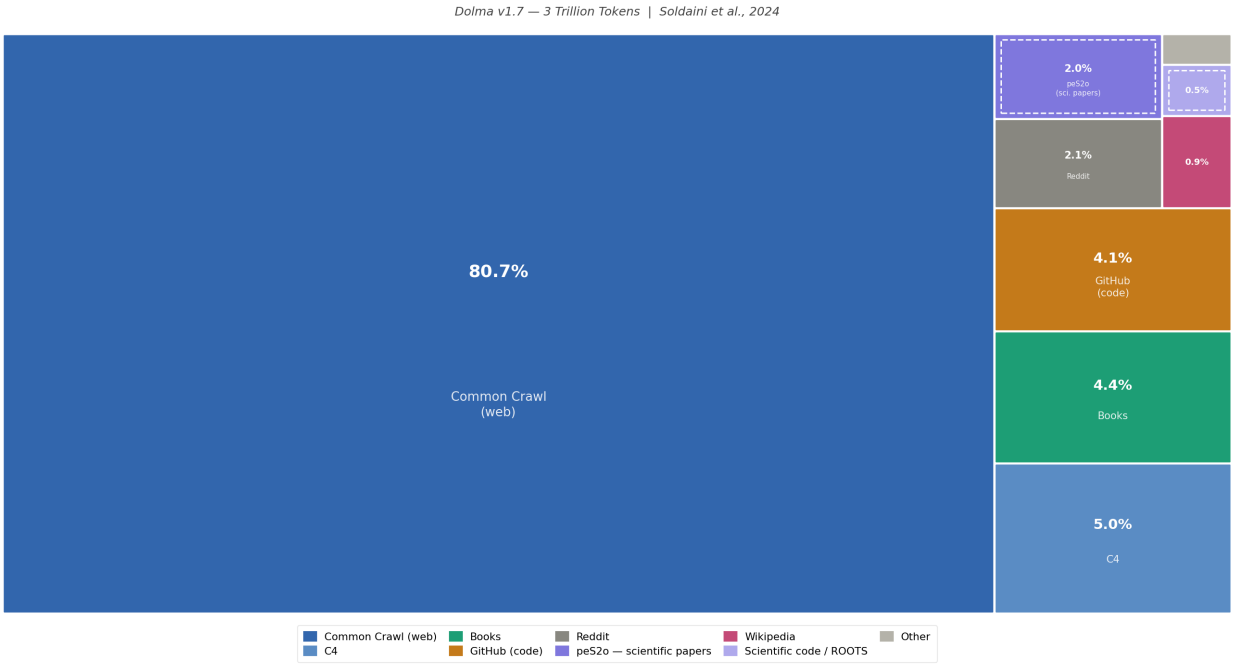

Figure 1.3: Token distribution across source corpora in Dolma v1.7, a 3-trillion-token open pretraining dataset [147]. Web crawl data (Common Crawl and C4) collectively account for 85.7% of all tokens, while academic scientific literature (peS2o [146]) and scientific code constitute approximately 2.5% combined (highlighted with dashed borders). This disparity reflects the breadth-over-depth bias inherent in web-scale pretraining corpora, motivating the need for retrieval-augmented approaches that can supply domain-specific scientific knowledge at inference time.

Implementing a RAG pipeline involves several steps. First, a curated corpus of documents needs to be obtained. As of writing, the Portable Document Format (PDF) is the dominant format for the dissemination of scientific literature. Since this format is not natively machine-readable, document parsing is necessary for extracting the raw textual or visual content from PDFs. All LLMs today feature a limited context length: the total number of tokens a model can process in a single forward pass is bounded by its architecture. Combined with inference costs in terms of latency and energy, this limitation calls for careful management of prompt size. In order to select only the most relevant content to answer a given query, the raw text obtained after document parsing must be broken down into smaller chunks.

Following the chunking stage, each chunk is encoded into a dense vector representation

by an encoder model. Crucially, the choice of encoder is not merely an implementation detail — it is the choice of data representation itself, as the encoder determines the geometry of the embedding space and therefore what "similarity" means at retrieval time. A chunk that is semantically relevant to a query can only be retrieved if the encoder maps both into proximate regions of that space; a poorly chosen encoder can render even a well-constructed corpus effectively unsearchable. In practice, this decision reduces to a fundamental trade-off between large, general-purpose encoders and smaller, domain-adapted ones. The former offer stronger generalization across diverse query types by virtue of training on broad corpora, while the latter offer specialization that may be critical in technical domains where terminology, notation, and reasoning patterns deviate substantially from general language. This trade-off carries significant consequences for downstream retrieval quality and should be treated as a primary design decision in any RAG pipeline, rather than a secondary concern left to default choices.

Embeddings produced by the encoder are stored in a vector database, also referred to as a vectorstore. A well-designed vectorstore is both lightweight — so that the data can fit into memory — and fast, achieved through highly efficient low-level kernels that accelerate nearest neighbor lookups based on vector distance. In this work, we employ FAISS [41], a state-of-the-art vectorstore in which each embedding is associated with a unique `uint64` identifier that maps to the raw content of its corresponding text chunk. When a user submits a query, it is encoded with the same encoder used for encoding the document chunks. The query embedding is then used to perform a semantic search in the vectorstore, retrieving its nearest neighbors which, assuming proper upstream chunking and encoding, are the most relevant chunks for answering the query.

The final step of the RAG pipeline involves feeding the question, retrieved context, and a surrounding prompt template with system instructions to a generator LLM. It is important to note that none of the aforementioned steps alter the weights of the generator LLM;

therefore, the retrieval and generation components of a RAG pipeline can be considered fully decoupled from one another. The LLM then answers the query drawing on both its parametric knowledge acquired during pretraining and the retrieved context. Contextualizing the query with retrieved evidence enables in-context learning [24], whereby the LLM's effective knowledge can extend beyond the boundaries of its pretraining corpus.

1.4 Challenges in Scaling RAG for Science

Despite its promise, scaling RAG to handle millions of scientific documents introduces significant challenges. The complexity of the RAG pipeline—spanning document parsing, text-chunking, encoding, retrieval, and generation— creates multiple potential points of failure that become acute at scale:

Parsing scientific documents. The Portable Document Format (PDF) has become the primary mode of scientific communication [5]. Scientific documents present particular difficulties due to their dense arrangements of tables, figures, equations, and references. Extraction-based parsers rely on embedded text layers that are only present in digital-born PDFs while end-to-end neural approaches face scalability challenges due to the quadratic scaling of vision transformer attention mechanisms. Developing a document parser that is simultaneously accurate, multimodal, and scalable to millions of papers is a prerequisite for any large-scale scientific RAG system.

Optimizing accuracy for scientific content. The quality of a RAG system's output depends critically on the encoder's ability to produce vector representations that capture the semantic nuances of scientific text. General-purpose encoders perform suboptimally on scientific content, as such content constitutes only a small portion of mainstream LLM training datasets [148, 47]. Academic writing style differs substantially from the web-scale

text on which most encoders are trained, and domain-specific vocabulary, empirical evidence, and citation patterns require specialized encoding strategies.

Evaluating RAG on scientific literature. Scientific literature carries unique features: domain-specific terminology, quantitative empirical evidence, and citation structures. Taken together, these idiosyncrasies of scientific text impose restrictions on how RAG systems can be evaluated. Existing general-purpose benchmarks are increasingly contaminated by model pretraining corpora [88, 37], rendering them unreliable indicators of true scientific reasoning ability. The development of fresh, provenance-tracked, science-specific benchmarks is essential for driving progress in this area.

Scaling to high-performance computing infrastructure. Indexing millions of scientific documents requires computational resources that exceed the capacity of conventional computing infrastructure. Each stage of the RAG pipeline (parsing, chunking, encoding, and retrieval) must be parallelized across thousands of accelerators on high-performance computing clusters. This introduces substantial engineering challenges around distributed workflows, GPU utilization, I/O bottlenecks that must be addressed to realize the vision of a computable scientific corpus.

1.5 Thesis Statement and Contributions

This thesis presents the design, implementation, and evaluation of retrieval-augmented reasoning systems that operate at the scale of millions of scientific documents, leveraging exascale supercomputing infrastructure to transform how scientists interact with the growing body of scientific literature. The central claim of this work is:

> *The path to reliable scientific AI lies not in larger models but in co-designing retrieval, reasoning, and HPC infrastructure as a unified system.*

1.6 Organization of This Thesis

The remainder of this thesis is organized as follows:

Chapter 2 surveys the state of the art in large language models, retrieval-augmented generation, document parsing, neural information retrieval, and scientific benchmarking, establishing the technical foundations upon which this work builds.

Chapter 3 presents HiPerRAG, the scalable RAG infrastructure that constitutes the first major contribution of this thesis. This chapter details the design and implementation of Oreo and ColTrast, introduces the scientific evaluation benchmarks, presents comprehensive results on scientific question-answering tasks and scaling experiments across three leadership-class supercomputers, and discusses downstream applications of HiPerRAG in multi-agent scientific discovery frameworks.

Chapter 4 presents the automated MCQA benchmark generation pipeline and the reasoning trace retrieval paradigm constituting the second major contribution.

Chapter 5 introduces Swarm Retrieval, the proposed paradigm for reasoning-driven document selection, and presents its design rationale, architecture, and evaluation framework.

Chapter 6 concludes the thesis with a discussion of limitations, broader implications, and future research directions.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter surveys the technical landscape upon which this dissertation builds. We organize the discussion around six themes: the evolution of large language models (Section 2.1), their application to scientific domains (Section 2.2), the development of retrieval-augmented generation and its variants (Section 2.3), document parsing for scientific literature (Section-Section 2.4), neural information retrieval and metric learning (Section 2.6), the evaluation of LLMs and RAG systems on scientific tasks (Section 2.7), and high-performance computing for AI workloads (Section 2.8). For each theme, we trace the arc from foundational contributions to the present state of the art, identifying the open problems that motivate the contributions of this dissertation.

2.1 Large Language Models

2.1.1 The Transformer Architecture

The Transformer architecture [158] replaced recurrent computation with multi-head self-attention, enabling parallelized training over long sequences and establishing the foundation for virtually all subsequent progress in language modeling. Given an input, sequence $X \in \mathbb{R}^{n \times d}$, the self-attention mechanism computes:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.1}$$

here $Q = XW^Q$, $K = XW^K$, $V = XW^V$ are linear projections into query, key, and value spaces and d_k is the key dimensionality. Multi-head attention maintains h parallel instances of this operation with the intuition that each attention head *attends to* different representation subspaces. Output from all heads are finally concatenated and fed through a final

projection W^O. Combined with position-wise feed-forward networks, layer-normalization [10], and residual connections [61], the Transformer block is stacked L times to form deep architectures whose parameter count can well exceed trillions as of writing.

Two variants have dominated subsequent development. Encoder-only models such as BERT [39] employ bidirectional attention and masked language modeling, producing contextualized representations suited for downstream natural language understanding (NLU) tasks such as classification, named-entity recognition, and retrieval. Decoder-only models such as the GPT family [127, 24] employ causal (sequential and one-way) attention and next-token prediction to power autoregressive generation wherein each token is sampled from a probability distribution conditioned on every preceding token. Decoder-only architecture emerged as the workhorse of the current wave of generative AI adoption. Beyond their generative capabilities, decoder-only models have also been fine-tuned as performant encoders [134]. In this thesis, we leverage both: ColTrast the encoder fine-tuning objective introduced in Chapter 3 applied to both encoder-only architectures like PubMedBERT [36] and decoder-only models like SFR-Embedding-Mistral [134].

2.1.2 Pretraining Data and the Scientific Domain Gap

The capabilities of modern LLMs derive in large part from the scale and diversity of their pretraining corpus. The Pile [47], and 800GB dataset spanning 22 sources, RedPajama [154] at trillion-token scale, and the Dolma [148] at three trillion tokens represent the current generation of open pretraining data. As the field runs out of novel human-generated training data, LLMs are utilized at scale to generate high-quality synthetic training corpora for themselves through techniques like rephrasing human-written text [116]. Despite their scale, most of these general-purpose datasets share a systematic preference towards breadth over depth, which has direct consequences for knowledge-intensive scientific applications. Ling et al. [91] find that scientific text constitutes only a small fraction of mainstream corpora.

The Pile, for instance, reportedly allocated approximately 14% of its composition to academic sources. The remaining content is web-text, code, and books, leaving general-purpose LLMs without sufficient exposure to unique token distributions that characterize scientific communications.

This gap of representation is not closed by expanding the scale alone. Scaling analyses [78, 65] establish that downstream model performance depends on the *composition* of training data as much as its sheer volume, i.e., a model trained on a trillion tokens of web text will not internalize scientific reasoning that was never in its training distribution. This is the primary motivation for retrieval-augmented generation approaches like ours that dynamically supplement foundation model's parametric memory with domain-specific evidence at inference time. This is the core principle of this dissertation: introducing the data into the system as an *extrinsic* reference source in contrast to continual pretraining and fine-tuning techniques that embed the data *intrinsically* in the parameters of the model. Crucially, the latter is costlier, more rigid, and prone to catastrophic forgetting [82, 53].

2.1.3 Reasoning in Language Models

Reasoning, in the context of LLMs, refers to a technique that urges the model to generate intermediate reasoning tokens preceding the final answer. Similar to RAG, this technique alters the conditioning context for next-token sampling—RAG by retrieval-augmented context, reasoning by generating intermediate computational steps. Chain-of-thought (CoT) prompting [167] demonstrated that eliciting intermediate reasoning steps substantially improved the performance on arithmetic, commonsense reasoning, and symbolic tasks. Self-consistency [163] extends this by sampling multiple reasoning paths and selecting the most common answer. Tree-of-thought [175] generalizes to tree-structured exploration with deliberate backtracking.

Most recently, reasoning-specialized models have demonstrated that structured reasoning

can be trained explicitly rather than invoked at inference time through prompting techniques. OpenAI's o1 [119] and DeepSeek-R1 [34] are trained with reinforcement learning (RL) over chain-of-thought data, producing models that generate extended *reasoning traces* before arriving at answers and achieving substantial gains on mathematical and scientific benchmarks. DeepSeek-R1 further demonstrated that reasoning can *emerge* unintentionally through RL that is aimed at optimizing downstream performance. In other words, the model internally re-invents reasoning as a strategy to maximize its performance on the RL task at hand. Additionally, and crucially for this dissertation, DeepSeek-R1 also demonstrated that these capabilities transfer to smaller models through fine-tuning-based distillation of the reasoning traces themselves.

The emergence of explicit reasoning traces has a direct connection to the contributions of this dissertation. In Chapter 4, we demonstrate that reasoning traces generated by frontier models can serve as not only supervised fine-tuning data, but also retrieval sources for smaller models, enabling them to approach or exceed frontier-level performance on domain-specific scientific tasks they were not explicitly trained for. This finding suggests that the value of reasoning lies not only in the model that produces it, but in the traces themselves as reusable, retrievable artifacts. However, reasoning traces are currently treated as a byproduct of the inference process and are discarded (unless specifically logged for future model training). Competition between frontier AI laboratories and utility of reasoning traces as distillation material play a key role in this systematic waste of valuable data, a waste we hope to raise awareness about by this dissertation.

2.1.4 Hallucination and Factuality

Despite their revolutionary capabilities, LLMs remain prone to hallucinate, i.e., to generate content that is cogent and seemingly plausible, yet factually incorrect [69]. Hallucination arises from a fundamental feature of LLMs as machine learning models: they sample from a

learned probability distribution to generate new content. In the autoregressive case, which is the backbone of the more dominant decoder-based architectures today, this sampling is conditioned on the prompt and all the other tokens theretofore generated. Since the next token is simply the most probable token, the LLMs offer no guarantee in its factuality, appropriateness, or topicality. In other words, the most probable next token is not always the most truthful one.

Huang et al. [67] distinguish intrinsic hallucinations (outputs that contradict a given reference material) from extrinsic hallucinations (outputs that cannot be verified against any source). Both are dangerous in scientific contexts, but the latter is harder to differentiate from a truly novel new deduction or hypothesis. An intrinsic hallucination might misrepresent an experimental result or violate a theoretical constraint while an extrinsic hallucination could fabricate a perniciously plausible nonexistent citation or fact. The severity of the problem in scientific applications is well-documented: GALACTICA [150] invoked tremendous excitement from the scientific community as the first purely scientific foundation model. Yet, the model was retracted from public access within three days of release due to its tendency to produce authoritative-sounding fabrications. Research has been done to quantify hallucination rates among LLMs: Min et al. [107] introduced FactScore, which decomposes generated text into atomic claims and verifies each against a known source, revealing that even GPT-4—the frontier LLM at the time— produced factually unsupported claims at non-trivial rates.

Several mitigation techniques have been proposed including post-training ones like DoLA [29], and training-based ones such as Reinforcement Learning from Human Feedback (RLHF) [122] and Direct Preference Optimization (DPO) [128]. We are of the opinion that the most intuitive, robust, and expedient approach remains grounding generation in externally retrieved attributable evidence. This is the core principle of retrieval-augmented generation, discussed in Section 1.3.

2.2 LLM-Based Systems for Scientific Discovery

The application of deep learning to scientific domains has followed multiple trajectories, each targeting a different modality of scientific data. It is important to distinguish between these often conflated trajectories, e.g., referring to protein structure prediction models as large language models. We present in this section a survey of the most relevant trajectories in which LLMs have been applied for facilitating scientific discovery.

2.2.1 Text-Based Scientific Language Models

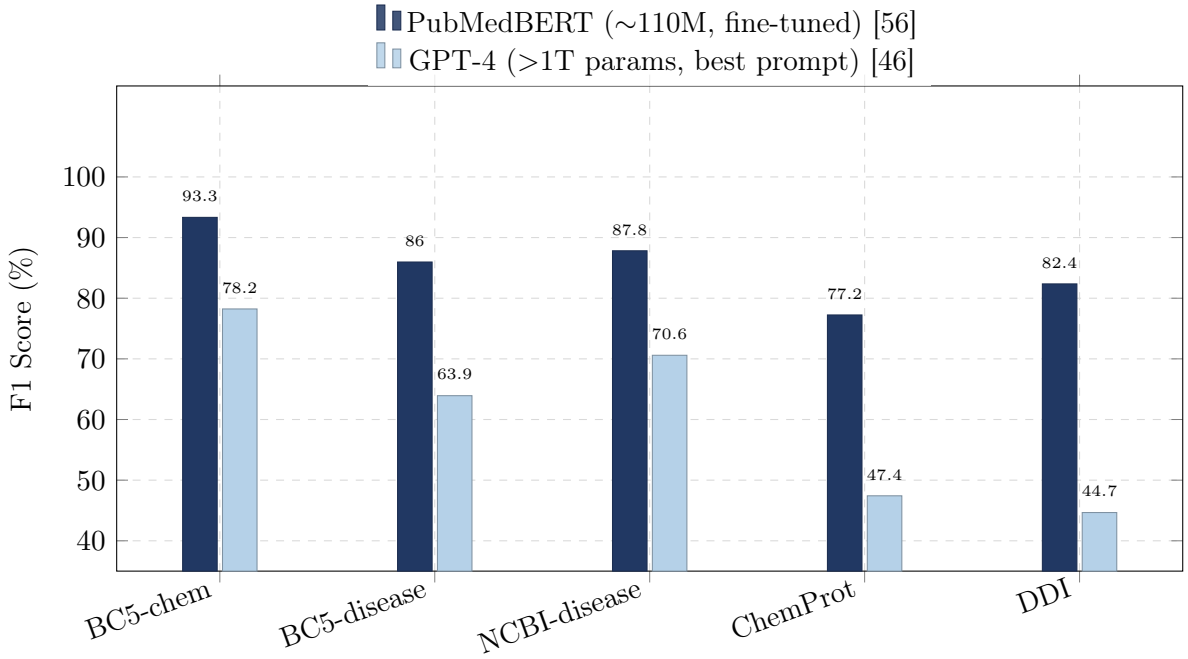

Figure 2.1: Performance of PubMedBERT [56] ($\tilde{1}$10M parameters, fine-tuned) versus GPT-4 (>1 trillion parameters, best prompt) on five BLURB biomedical NLU tasks [46]: three named entity recognition (NER) and two relation extraction (RE) datasets. Despite a parameter-count difference of roughly four orders of magnitude, the domain-specific encoder outperforms GPT-4 by 15.54–31.55 F1 points on NER and 24.78–41.46 points on RE.

The first trajectory adapts the Transformer architecture for scientific text. The key insight is that scientific language differs substantially from colloquial web text in vocabulary, syntax, discourse structure, and inference patterns. Models trained on general text underper-

form on scientific tasks not because they lack capacity, but because of insufficient exposure to scientific corpora. The divergence in token distributions between scientific and web-text corpora calls for customized tokenizers. Generic tokenizers, despite offering full vocabulary coverage, remain inefficient—requiring more tokens to encode scientific text than general text.

SciBERT [17], introduced by the Allen Institute for AI, pretrained a BERT-base model on 1.14 million papers from Semantic Scholar, yielding consistent improvements over vanilla BERT on scientific NLU tasks including named-entity recognition (NER), relation extraction, and classification. PubMedBERT [56] took this further by pretraining exclusively on PubMed abstracts and full-text articles, demonstrating that domain-specific pretraining from scratch can outperform continued pretraining and task-specific fine-tuning from a general checkpoint. Domain-specific encoders such as PubMedBERT have been observed to outperform their much larger generic counterparts in biomedical NLU (see Figure 2.1). However, the trade-off between larger, more generalizable but slower and more resource-intensive models and relatively lightweight, faster niche models remains an open research problem.

Table 2.1: Comparison of large generic language models versus lightweight domain-specific encoders across key evaluation criteria. ✓ indicates a clear strength, × a weakness, and ∼ a partial or context-dependent advantage.

Criterion	**Large Generic Models**	**Domain-Specific Encoders**
Out-of-domain generalization	✓	×
Computational efficiency	×	✓
Inference latency	×	✓
Pretraining data requirements	✓	∼
Fine-tuning flexibility	✓	∼
Breadth of applicability	✓	×
Vocabulary alignment to domain	×	✓

On the generative-side, BioGPT [101] trained a GPT-2 scale model on 15 million PubMed abstracts, achieving a state-of-the-art performance on biomedical text generation and relation extraction. BioMedGPT [102] extended this to a multimodal setting that integrated text, molecular structures, and protein sequences into a unified generative framework. GALACTICA [150] represented the most ambitious attempt at the time to build a general-purpose scientific foundation model (LLM), training on a corpus of 48 million scientific papers, textbooks, encyclopedias, and knowledge bases. While GALACTICA demonstrated impressive capabilities for the brief period for which it was deployed, its susceptibility to a self-confident hallucination on scientific claims led to its rapid withdrawal from public access. This incident raised awareness about the limits of parametric knowledge of LLMs for scientific applications.

Med-PaLM-2 [144] marked a milestone in the field by achieving human expert-level performance on the United States Medical Licensing Examination by combining the base PaLM model with biomedical-focused instruction fine-tuning. This result demonstrated that LLMs can reach professional certification thresholds on medical knowledge tasks, though the gap between examination performance and reliable clinical reasoning remains significant. Furthermore, there is alarming evidence that popular LLM evaluation benchmarks often leak into the pre and/or post training datasets of frontier LLMs, artificially boosting their scores.

Before we conclude this chapter, it is worth noting that the term "AI for science" encompasses architectures far beyond text-based language models and the scope of our work. Notable examples within the realm of biological/biomedical domains include protein structure prediction systems such as AlphaFold [75] and Proteina [48]; protein sequence models such as ESM-3 [60] and xTrimoPGLM [26]; and genomic language models such as GenSLMs [181] and Evo [23]. These systems operate on biological sequences and three-dimensional coordinates rather than natural language and are architecturally distinct from the text-based LLMs that the focus of this dissertation. We mention them here only to delineate the scope of our contributions: this dissertation addresses how LLMs interact with the *scientific litera-*

ture rather than the prediction of molecular structures or biological properties. Where these systems appear in our work (for example, AlphaFold-2 appears as a tool alongside HiPerRAG chapter 3 in the StructBioReasoner agentic framework) they do so as downstream consumers of the retrieval infrastructure we develop, not as our objects of study in themselves.

2.2.2 AI Agents for Scientific Discovery

The most recent trajectory in AI for science shifts focus from standalone models to agentic systems that orchestrate multiple AI models and computational tools within a reasoning and planning loop with constituent long and short-term memory constructs. This paradigm recognizes that scientific discovery is not a single-model task but a multi-step pipeline that involves hypothesis generation, experimental design, data analysis, iterative refinement, and dissemination.

Boiko et al. [19] demonstrated Coscientist, a semi-autonomous chemical research agent capable of searching the literature, planning synthesis routes, writing code for robotic platforms, and executing experiments. Lu et al. [99, 100] developed *The AI Scientist* (Sakana AI), which fully automates the entire research lifecycle from idea generation through manuscript composition and peer review, with papers exceeding acceptance thresholds at top machine learning venues. Mitchener et al. [108] introduced Kosmos, an AI scientist for data-driven discovery that uses structured world models to coherently pursue research objectives over extended campaigns. A single Kosmos run is reported to produce approximately 42,000 lines of code and synthesize over 1,500 papers, with independent scientists validating 79.4% accuracy and reporting that one run compressed six months of expert work into a single day. More recently, Lyu et al. [103] introduced EvoScientist, an evolving multi-agent framework that continuously improves research strategies through persistent memory and self-evolution, comprising specialized agents for idea generation, experiment implementation, and knowledge distillation, representing the current state-of-the-art in agentic AI for science.

A common architecture underpins the aforementioned agentic systems: an LLM-based reasoning core that plans actions, invokes tools, interprets results, and directs next steps. Retrieval-augmented generation is one of the most critical tools available to LLMs serving this role as it provides access to the scientific literature for evidence gathering and hypothesis contextualization. The StructBioReasoner system [141], discussed in chapter 3, examplifies this architecture in the domain of biologics design, with HiPerRAG serving as the literature synthesis agent within a multi-agent tournament framework.

The emergence of agentic scientific AI underscores the central argument of this dissertation: the value of LLMs for science lies not in their standalone generation capabilities, but in their capacity to serve as reasoning engines within larger systems that include retrieval, simulation, and in-vitro experimental feedback loops. Building reliable scientific agents requires reliable retrieval, and reliable retrieval at scale requires the kind of HPC infrastructure this dissertation develops. To conclude, our aim extends beyond developing an interface for *human scientist* to interact with the entirety of the science that has been disseminated so far, but also building that interface for the *AI scientists* of today and tomorrow.

2.2.3 Knowledge Distillation and Reasoning Transfer

It is well-established that scale of LLMs in parameter count and training data often dictates their performance [65, 78]. A recurring challenge in deploying LLMs for scientific applications is that the most capable *frontier* models are also the most expensive to run. Frontier models such as GPT-5.5 and Claude Opus 4.7 deliver strong reasoning but impose substantial per-token costs, latency constraints, and data-sovereignty concerns that make them impractical for many scientific workflows –particularly agentic settings that require thousands of LLM calls per campaign. Smaller models –which, as of writing, correspond to models with roughly 1-14 billion parameters– are cheaper, faster, and deployable local infrastructure, but they lack the reasoning depth and domain knowledge of larger counterparts. This tension motivates

knowledge distillation: transferring the capabilities for a large *teacher* model into a smaller *student* model.

Classical Knowledge Distillation

The foundations of knowledge distillation were laid by Hinton et al. [63] in their seminal paper where they showed that a student network can be trained to mimic the soft probability distribution (dark knowledge, as Hinton calls it) produced by a teacher network, rather than training on hard labels alone. The teacher's softened output distribution encodes inter-class similarities that the hard labels discard, e.g., that a given protein sequence is more similar to kinases than to membrane receptor, even though both are incorrect labels for the specific query. By training the student to reproduce these soft targets, the teacher's relational knowledge is transferred into a smaller architecture.

Formally, given a teacher model T producing logits z_T and a student model S producing logits z_S, the distillation loss is:

$$\mathcal{L}_{\text{KD}} = \alpha \cdot \mathcal{L}_{\text{CE}}(y, \sigma(z_S)) + (1 - \alpha) \cdot \tau^2 \cdot D_{\text{KL}}(\sigma(z_T/\tau) \,\|\, \sigma(z_S/\tau)) \tag{2.2}$$

where σ denotes the softmax function, τ is a temperature parameter that controls the smoothness of the distributions, y is the ground-truth label, and α balances the task loss against the distillation loss. This formulation has been remarkably effective across domains, from image classification [63] to language understanding [135, 73].

DistilBERT [135] applied this approach to compress BERT into a model 40% smaller and 60% faster while retaining 97% of its language understanding capability. TinyBERT [73] extended distillation to intermediate transformer layers, transferring not just output distributions but also attention patterns and hidden representations. More recently, Ye et al. introduced Generative Adversarial Distillation (GAD), which enables black-box distil-

lation of large language models by framing the student LLM as a generator and training a discriminator to distinguish its responses from the teacher's, creating an adaptive on-policy reward mechanism that avoids the exposure bias and overfitting to local patterns inherent in traditional sequence-level knowledge distillation [176]. These works established that Transformer-based language models are amenable to distillation, but they operated on representations and logits, not on the reasoning processes that produce those outputs. This is a research gap we aimed to target in this dissertation.

From Output Distillation to Reasoning Distillation

The advent of chain-of-though (CoT) reasoning subsection 2.1.3 opened a new dimension of distillation: transferring not just *what* a model eventually predicts, but also *how* it reaches that final prediction. The key observation is that an LLM's chain-of-thought trace encapsulates a structured reasoning process wherein the model identifies relevant principles, decomposes the problem, eliminates incorrect hypotheses, and builds towards a conclusion through in-context self-reflection. This reasoning trace, in and of itself, represents a valuable training signal for a smaller model. In fact, we show that even in the lack of a teacher's final predictions, student models can learn to reason towards the correct choice solely based on distillation from a teacher's reasoning traces.

Mukherjee et al. [111] demonstrated this with Orca, which pretrains a 13B-parameter student model on the detailed reasoning traces produced by GPT-4. Rather than simply mimicking the teacher's final answers, the student learns to reproduce the teacher's reasoning process step by step. Orca achieved performance rivaling ChatGPT on several benchmarks, despite being substantially smaller, because the reasoning traces provided a richer and denser training signal than answer labels alone. Subsequent work on Orca-2 [109] refined this approach by teaching the student to use different reasoning strategies depending on the tasks, demonstrating that reasoning distillation can transfer not just knowledge but metacogni-

tive strategy. Our work in Chapter 4 builds on this approach by investigating the impact of different types of reasoning traces (step-by-step, recall-then-generate, direct answer) as retrieval sources to assist scientific multiple-choice questions-answering of small models.

Ho et al. [64] provided a systematic study of fine-tuning small models on chain-of-thought rationales generated by large models, demonstrating that this approach enables small models to perform multi-hop reasoning tasks where they previously failed. Magister et al. [104] extended this to a broader range of reasoning tasks, finding that chain-of-thought distillation consistently improves small model performance on arithmetic, commonsense, and symbolic reasoning benchmarks.

More recently, the most striking demonstration of reasoning distillation at scale came from DeepSeek-R1 [34]. Following a ground-breaking multi-step recipe that is out of the scope of our dissertation, DeepSeek first trained a large reasoning model (DeepSeek-R1, 671B parameters). Remarkably, they showed that reasoning can *emerge* inadvertently as a by-product of reinforcement learning for accuracy and format-following. In other words, the model had what the DeepSeek team called an "aha moment" where it realized that reasoning on a problem leads to improved downstream RL reward even though the award itself did not measure reasoning capability. DeepSeek-R1 was used to produce extended reasoning traces, which are then distilled into smaller models (1.5B to 70B parameter Qwen and Llama variants) through supervised fine-tuning. The distilled models achieved notable performance: the 14B distilled Qwen model outperformed the GPT-4o and Claude3.5 Sonnet (both capable proprietary models at the time) on several mathematical reasoning benchmarks. This result demonstrated that reasoning capabilities, once elicited in a large model, can be effectively compressed into much smaller architectures through trace-based distillation.

The Missing Modality: Distillation Through Retrieval

We previously mentioned that introducing new data and capabilities to a model can be done in two different ways: intrinsically or extrinsically (subsection 2.1.2). All of the approaches described above share a common mechanism, they transfer knowledge by modifying the student model's weights, integrating new data as an intrinsic element of the model. Whether the training signal is soft logits (Hinton), intermediate representations (TinyBERT), or reasoning traces (Orca, DeepSeek-R1-Distill), the distillation process produces a new model whose parameters encode the transferred knowledge.

This weight-based distillation paradigm has two limitations for scientific applications. First, it requires a separate distillation procedure for each target domain: a student model distilled on biomedical reasoning traces may not transfer to material science or astrophysics, necessitating repeated, expensive, and error-prone distillation efforts. Second, the distilled knowledge is static, i.e., frozen into the student's weights at training time, and cannot be updated as the scientific literature evolves without further retraining.

chapter 4 of this dissertation introduces an alternative mechanism that we term *distillation through retrieval*. Rather than using reasoning traces to modify the student's weights, we store the traces as retrievable artifacts in a vector index. At inference time, the student model retrieves relevant traces and uses them as in-context examples, gaining access to the teacher's reasoning without any weight modification. This approach is model-agnostic (any student model can benefit after the creation of the index), domain-flexible (traces from different domains can be stored in the same index), dynamic (the contents of the index can evolve with the underlying scientific discourse), and transparent (traces are human-readable and auditable). To our knowledge, this represents the first systematic exploration of retrieval as a distillation mechanism for scientific domain adaptation.

The conceptual arc from soft target distillation [63], to representation distillation [73], to reasoning-trace distillation [34, 111], to our proposed distillation through retrieval rep-

resents a progressive shift from intrinsic to extrinsic knowledge transfer as we define in subsection 2.1.2. At each step, the transferred artifact becomes more interpretable and more reusable, culminating in reasoning traces that can be inspected, curated, and dynamically composed at inference time.

2.3 Retrieval-Augmented Generation

2.3.1 Foundations: Dense Passage Retrieval

Modern retrieval-augmented generation (RAG) builds on the foundation of dense passage retrieval (DPR), introduced by Karpukhin et al. [79]. DPR employs a dual-encoder architecture in which separate BERT-based encoders produce dense vector representations for queries and passages. Given a query q and a set of passages $\{d_1, d_2, \ldots, d_N\}$, the encoders produce representations $\mathbf{h}_q = E_Q(q)$ and $\mathbf{h}_{d_i} = E_D(d_i)$, and relevance is computed as the dot product $\text{sim}(q, d_i) = \mathbf{h}_q^\top \mathbf{h}_{d_i}$.

The dual-encoder architecture affords a crucial computational advantage: document representations can be precomputed and stored in an index, so that at query time, only the query needs to be encoded. The nearest-neighbor search over the precomputed document representations can then be performed efficiently using approximate neares-neighbor (ANN) libraries such as FAISS [41], which implements several index types including flat (exact) search, inverted file (IVF) indices, and product quantization (PQ) for compressed representations.

DPR was a seminal work because it demonstrated that learned dense representations substantially outperform traditional sparse retrieval methods such as BM25 [133] on open-domain question answering tasks. This result established the dual-encoder paradigm as the standard approach for neural information retrieval and motivated the development of more sophisticated encoding and training strategies, including the ColTrast objective introduced

in this dissertation.

The training of dual encoders for DPR requires a dataset of query-passage pairs with relevance labels. DPR uses contrastive learning with in-batch negatives: for a batch of B query-positive pairs, each positive passage for one query serves as a negative example for all other queries in the batch, yielding $B - 1$ negative examples per query without additional computation. The quality of training depends critically on the selection of hard negative examples, i.e., passages that are semantically similar to the query but not relevant [171]. We discuss contrastive learning objectives in detail in Section subsection 2.6.3.

2.3.2 The RAG Framework

Building on DPR, Lewis et al. [87] introduced Retrieval-Augmented Generation (RAG), which combines a pretrained retriever with a pretrained sequence-to-sequence generator. The original RAG framework as its defined by Lewis et al. features two variants:

- **RAG-Sequence**: The same retrieved document is used to generate the entire output sequence. The model marginalizes over the top-K retrieved documents:

$$p(y|x) \approx \sum_{z \in \text{top-}K} p(z|x) \cdot p(y|x, z) \tag{2.3}$$

  where x is the input query, z is a retrieved document, and y is the generated output.

- **RAG-Token**: Different tokens in the output can attend to different retrieved documents, allowing the model to synthesize information from multiple sources within a single generation:

$$p(y_i|x, y_{1:i-1}) \approx \sum_{z \in \text{top-}K} p(z|x) \cdot p(y_i|x, z, y_{1:i-1}) \tag{2.4}$$

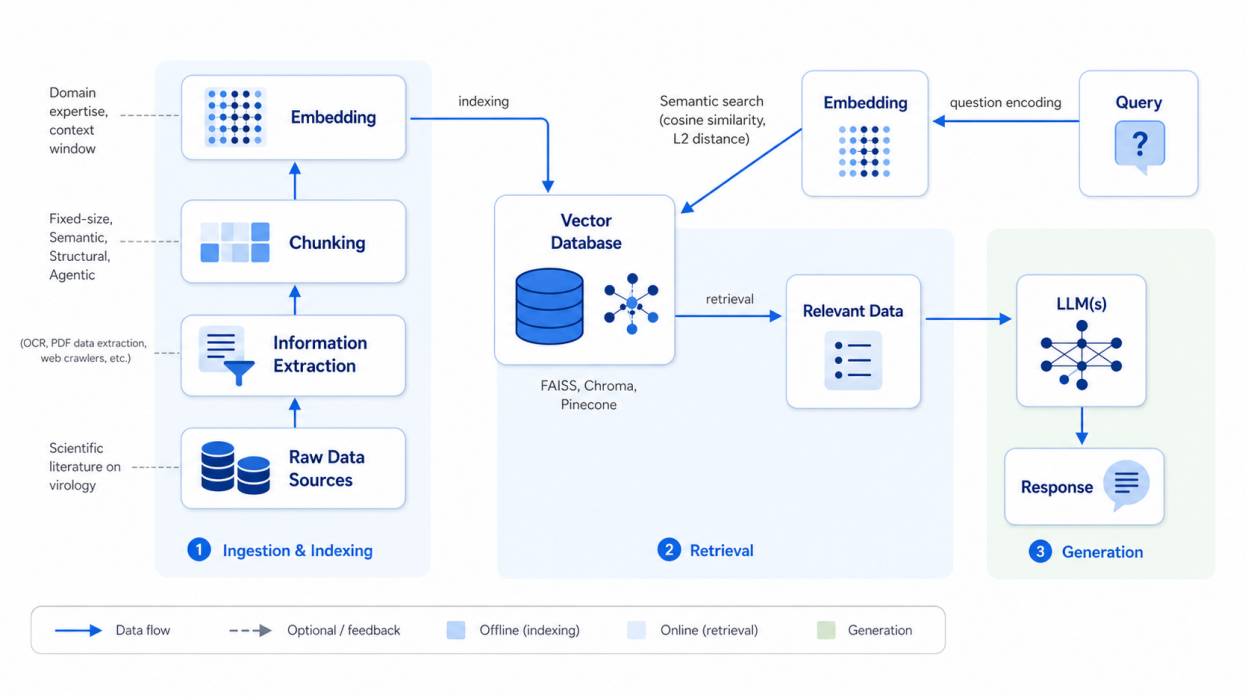

Figure 2.2: An end-to-end illustration of the retrieval-augmented generation (RAG) pipeline from data ingestion to response generation.

The RAG-Sequence variant runs substantially faster but suffers from coarse-grained context selection, which includes or discards entire documents. In contrast, RAG-Token operates more slowly because it considers tokens across documents, yet this cross-document referencing yields notably higher downstream accuracy on question-answering tasks.

The key insight of RAG was that this architecture could be trained end-to-end, with the retriever's selections influencing the generator's outputs. However, in practice, most modern implementations of RAG use a frozen retriever and frozen generator, with the retrieved passages simply concatenated with the query as input to the generator LLM. This retrieve-and-generate paradigm, while less custom-tailored to a given task than end-to-end training, is more practical at scale as it allows the retriever and the generator to be developed and deployed independently.

Concurrently with RAG, Guu et al. [58] proposed REALM (Retrieval-Augmented Language Model Pre-Training), which integrated retrieval directly into the language model pre-

training objective. REALM treats the retriever as a latent variable and trains it jointly with the language model, conceiving that what a model retrieves during pretraining shapes what it learns. This idea that retrieval can serve as a fundamental component of language model design rather than a mere inference-time augmentation has influenced subsequent architectures including RETRO [21], which chunks the training corpus and retrieves nearest-neighbor chunks during pretraining.

2.3.3 Advances in RAG Architectures

Since its inception, the RAG paradigm has evolved considerably along several dimenstions. We organize these advances by the type of limitation they address.

Adaptive retrieval. A key limitation of vanilla RAG is that it retrieves for every query, even when the model's parametric knowledge is sufficient to answer correctly. Self-RAG [8] addresses this by training the model to generate special 'reflection' tokens that indicate whether retrieval is needed, whether the retrieved passages are relevant, and whether the generated output is supported by the retrieved evidence. This self-reflective mechanism allows the model to decide when to retrieve and to reflect on the quality of its own retrieval-augmented outputs.

Active RAG [72] takes a complementary approach, using the model's uncertainty (measured by entropy over generated tokens) as a signal for when to trigger retrieval. When the model is confident, it falls back to its parametric memory (pretraining); when uncertain, it pauses to retrieve relevant evidence before continuing generation.

Corrective retrieval. Corrective RAG (CRAG) [173] mitigates the problem of low-quality retrieval by incorporating a lightweight evaluator that assesses the relevance of retrieved documents. When the evaluator determines the retrieved documents are not of sufficient relevance to the query, CRAG triggers a web search as a fallback mechanism, enabling the

system to self-correct retrieval errors. Albeit brittle in its reliance on keyword-based web searches, the system recognizes that no single local retrieval system can serve all queries and that graceful fallback mechanisms are essential for topical coverage.

Structured Retrieval. Vector-based vanilla RAG can be viewed as a greedy retrieval strategy, in which the top-K nearest neighbors are selected based on local similarity scores, without guaranteeing global relevance to the query. GraphRAG [43] initiated a departure from flat vector retrieval by constructing a hierarchical knowledge graph from the source corpus. The graph captures entities, relationships, and community structures within the text, enabling retrieval strategies to traverse the graph structure based on deterministic and interpretable heuristics. For queries requiring synthesis across multiple documents or multi-hop reasoning (e.g., "What proteins in the protein family Y interact with the binding partners of X?"), graph-based retrieval offers a natural alignment with the relational structure of scientific knowledge. Furthermore, thanks to its hierarchical layout, GraphRAG outperforms vanilla RAG in corpus-level queries that require a holistic perspective rather than specific details (e.g., What are the common structural vulnerabilities of this class of materials?). Knowledge graphs can be constructed from unstructured data like scientific papers using LLMs for named-entity extraction and relation tuple identification. Relational databases, meanwhile, integrate seamlessly with GraphRAG indexing approaches. A particularly novel variant of graph-based retrieval, HippoRAG [57], extended this idea by drawing inspiration from the hippocampal memory system, implementing a neurobiologically motivated indexing and retrieval architecture.

Agentic retrieval. The most recent wave in RAG is the integration of retrieval into agentic frameworks [142]. In agentic RAG, an LLM-based agent orchestrates retrieval as one tool among many within a broader reasoning loop, enabling it to reformulate queries, evaluate retrieval and generation performance, and adapt its retrieval strategy iteratively. In essence,

agentic retrieval provides a unified framework that synthesizes all prior advances in RAG evolution.

The ReACT framework [174] demonstrated that LLMs can learn to interleave reasoning with tool use, generating that the authors called *Thought-Action-Observation* traces that alternate between internal reasoning, external actions (taken through tool calling), and processing of results. Novelty of ReACT lies in its fidelity to how humans conduct scientific inquiry. Subsequently, ToolFormer [137] showed that LLMs can learn when and how to invoke tools being fine-tuned on self-generated tool-use annotations.

These developments lay the groundwork for retrieval as an active, agent-directed process rather than a passive lookup. Yet agentic RAG, as currently conceived, still treats documents as inert objects to be retrieved, i.e., the agency remains in the orchestrating agent, not in the documents themselves. Our Swarm Retrieval proposal (chapter 5) pushes this logic further: by embedding agency directly in the documents as *Paper Spirits*, we reconceive retrieval not as search but as distributed reasoning. Documents do not await selection; they argue for their own relevance, and judges adjudicate among their reasoned cases. In this paradigm, retrieval emerges from the data rather than being imposed upon it.

2.3.4 RAG for Scientific Applications

The application of RAG to scientific domains has yielded a growing body of work driven by the recognition that scientific question answering requires evidence-grounded responses that general-purpose LLMs cannot reliably provide from parametric memory alone.

PaperQA [84] demonstrated that retrieval over full-text scientific papers can enable LLMs to answer research-level questions with citations. PaperQA-2 [145] extended this with an agentic architecture that iteratively searches, gathers evidence, and generates cited responses, achieving performance approaching human experts on the LitQA benchmark (granted, LitQA benchmark originates from the same research group). OpenScholar [9] sub-

sequently demonstrated that a smaller open-weight model (8B parameters) augmented with a corpus of 45 million open-access papers and trained retrievers outperforms both PaperQA-2 and GPT-4o on ScholarQABench [4], achieving substantially lower cost by replacing proprietary LLM reranking with efficient bi-encoder and cross-encoder pipelines. These systems share a common finding: iterative, agent-directed retrieval with query reformulation substantially outperforms single-pass retrieval for complex scientific queries.

In the biomedical domain, BioASQ [155] established a series of annual challenges for biomedical semantic indexing and question answering that have served as standard evaluation benchmarks for over a decade. More recently, ScholarQABench [4] introduced a benchmark specifically designed to evaluate multi-paper scientific question answering, reflecting growing recognition that real scientific queries often require synthesizing evidence across multiple documents of varying formats.

Despite this progress, a persistent gap remains between the scale at which RAG systems are evaluated (in the order of hundreds to a few thousands of documents) and the scale at which they must operate to be scientifically useful (millions to hundreds of millions of documents). PaperQA 2 and OpenScholar, for instance, both retrieve over dynamically assembled corpora of at most a few hundred papers per query. Bridging this gap to millions of preindexed documents requires not only algorithmic innovations in retrieval accuracy but also systems-level engineering on high-performance computing infrastructure. This is the central focus of this dissertation.

2.4 Document Parsing for Scientific Literature

The extraction of machine-readable text from scientific PDFs represents the first and often most error-prone stage of any scientific RAG pipeline. The Portable Document Format (PDF) was designed for visual fidelity across devices, not for semantic accesibility [5]. Alongside the raw content, a PDF file stores rendering instructions such as character positions,

font specifications, drawing commands, etc. Counter-intuitively, the format does not feature metadata on logical structure such as paragraphs, sections, citations, etc. As a result, content extraction from PDFs poses a fundamentally underdetermined inverse problem. Scientific documents only exacerbate this challenge with their dense arrangements of multi-column layouts, mathematical notation, tables, figures, captions, footnotes, and reference lists.

2.4.1 Extraction-Based Parsing

Extraction-based parsers such as PyPDF [3], PyMuPDF [2], and PDFMiner [1] leverage the internal structure of PDFs to obtain text and metadata which they often output as structured text formats like XML. These tools access the text layer embedded within digital-born PDFs which are documents were created digitally rather than scanned from paper. Extraction-based parsing is CPU-bound and achieves high throughput–in our work, we observed typically thousands of pages per second on a single CPU– because it involves only string manipulation without GPU-bound neural network inference.

However, extraction-based approaches suffer from several innate limitations. First, they rely on the presence of an embedded text layer, which is absent in scanned PDFs and may be incomplete or corrupted even in digital-born PDFs. Second, their accuracy is sensitive to formatting errors within PDFs. These errors include, but are not limited to, misencoded fonts, incorrect chapter mappings, and inconsistent whitespace–all of which can produce distorted output. Third, extraction-based parsing is largely restricted to raw text and provides little or no support for tables, figures, equations, and code blocks, which constitute an irreplaceable portion of scientific content.

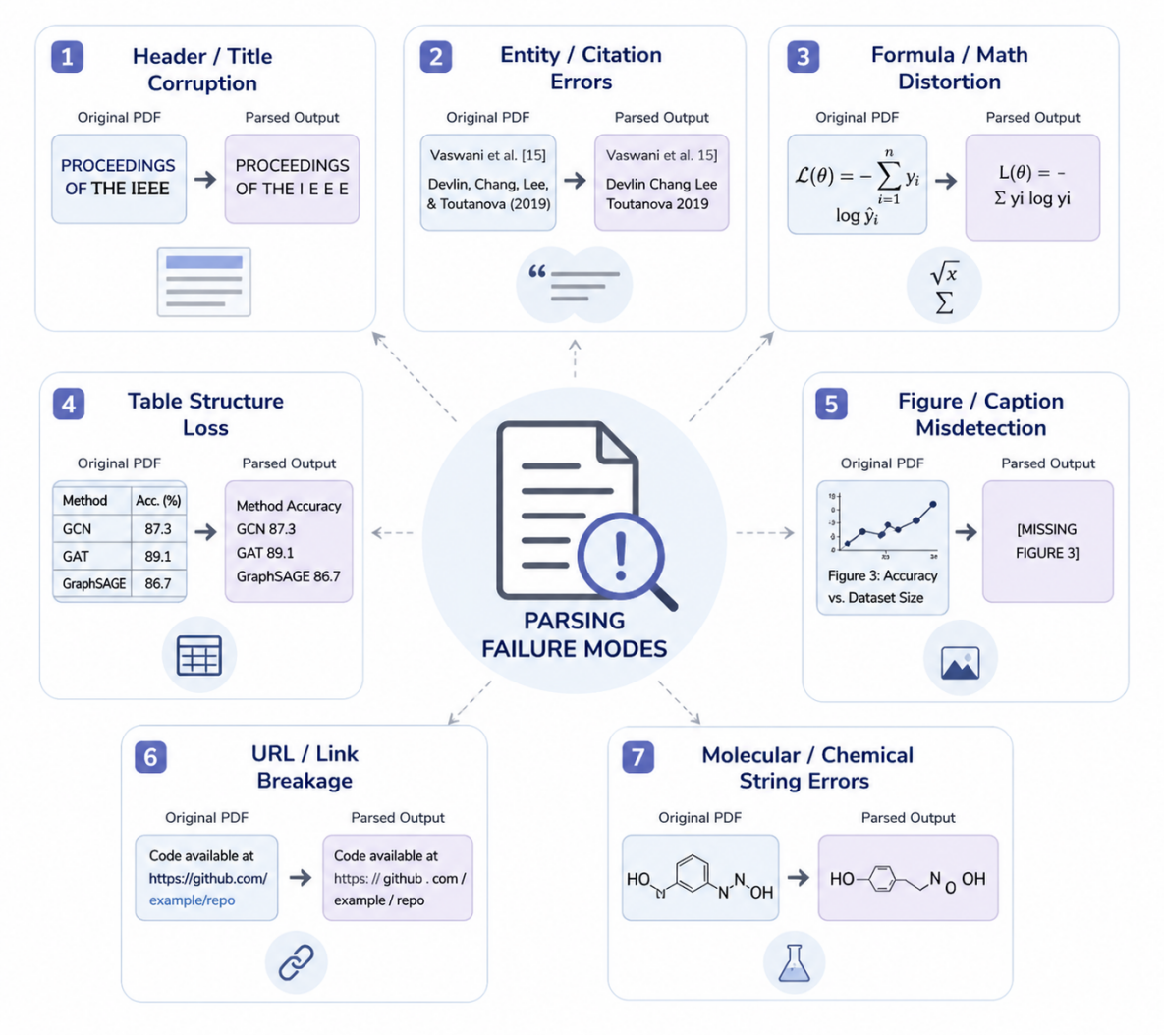

Figure 2.3: Common failure modes in scientific context extraction from PDFs.

2.4.2 Neural Document Parsing

The advent of deep learning in computer vision has enabled a new generation of document parsers that operate on the page images rather than PDF internals, thereby handling both born-digital and scanned documents.

Nougat [18] (Neural Optical Understanding for Academic Documents), developed by Meta, established the paradigm of end-to-end neural document parsing for scientific literature. Nougat uses a Swin Transformer [97] encoder to embed document page images and

a mBART-based [95] text decoder to generate Markdown output, including mathematical equations in LaTeX notation. By training on paired (page image, Markdown) data derived from LaTeX sources on arXiv, Nougat learns to produce structured text that faithfully represents the content of complex scientific documents.

Despite its strong performance, Nougat's end-to-end architecture encodes entire pages through the Vision Transformer's quadratic attention mechanism which results in significant computational overhead and limits the throughput to approximately 5-10 seconds per page on an NVIDIA A100 GPU. At the scale of millions of documents, that throughput becomes prohibitive without massive parallelization. More fundamentally, the monolithic design processes all page elements through the same architecture, failing to exploit the fact that different element types may benefit from specialized processing strategies. Subsequent evaluations have confirmed that Nougat underperforms on isolated elements such as tables and mathematical expressions when compared against modular alternatives [121].

Marker [33] takes a hybrid approach that combines extraction-based methods for simple text regions with neural methods for complex elements. In its current form, Marker supports a broad range of input formats and element types including tables, equations, inline mathematics, code blocks, etc. However, Marker's pipeline remains optimized for single-document throughput rather than the massive batch-processing regime required for corpus-scale scientific RAG.

As of writing, broader community has increasingly converged on *modular pipeline* architectures that address Nougat's throughput and coverage limitations simultaneously. Recent tools such as MinerU [160] and Docling [98] exemplify this approach by routing each detected layout region (such as text, tables, figures, mathematical equations) to a specialized submodel with those sub-models executing in parallel. In response to the need for a standardized evaluation benchmark for document parsing, Ouyang et al presented OmniDocBench [121] which covers 1,651 PDF pages across 10 document types with 28 block-level annotation

categories.

Despite these advances, no existing parser is simultaneously optimized for throughput at supercomputing scale, multimodal coverage across the full range of scientific element types, and adaptability to the heterogeneous quality of real-world scientific corpora. Moreover, these approaches do not embody a scale-oriented design that aims to saturate thousands of GPUs to achieve a throughput of thousands of PDFs per minute.

2.4.3 Adaptive and Hybrid Parsing

The observation that different documents require different parsing strategies motivated the development of adaptive approaches. Siebenschuh et al. developed AdaParse [140], which features a trained parser-selection model via direct preference optimization (DPO) on scientist preferences over competing parser outputs. At inference time, AdaParse routes each document to the parser most likely to produce high-quality output, achieving a 17× throughput improvement over state-of-the-art parsers at 0.2% better accuracy on a diverse, manually-curated benchmark of 1,000 scientific documents. The system runs on the Polaris supercomputer using Parsl for workflow orchestration, demonstrating adaptive parsing at leadership-class computing scale.

The Oreo parser introduced in this dissertation (Chapter 3) takes a complementary approach: rather than selecting among parsers, Oreo decomposes the parsing pipeline into a fast CNN-based layout detection stage and a targeted ViT-based text recognition stage, applying neural inference only where required. This compartmentalized design achieves both high throughput and broad multimodal coverage.

2.5 Text Chunking Strategies

Once raw text has been extracted from scientific documents, it must be segmented into chunks suitable for encoding and retrieval. This necessity is based on two equally important

factors. First, the finite context lengths of LLMs require careful selection of the content that will be included in the prompt. Even though recent advances in efficient attention techniques [32, 31] have paved the path for models with over a million-token context lengths [49, 106], certain scientific applications may still exceed these limits. Second, even when context limits are not strictly exceeded, longer inputs do not guarantee better performance. Liu et al. [94] reported a peculiar phenomenon where LLM accuracy degrades significantly when relevant information appears in the middle of a long context rather than at its boundaries. Coined *lost in the middle phenomenon*, this degradation persists even under conditions of perfect oracle retrieval [42]. Figure 2.4 illustrates an unintended but informative result from our own experiments that further supports the lost-in-the-middle phenomenon. In this study, we examined how different encoders influence the cutoff boundaries used in semantic chunking. Due to a misconfigured hyperparameter in ModernBERT [164], entire documents were collapsed into single chunks. Consequently, the retrieval stage returned a small set of full-length scientific papers (e.g., $K = 10$) rather than targeted passages. This shift in retrieval granularity led to a consistent and substantial degradation in performance across all evaluated LLMs. These findings motivate careful curation of prompt content rather than indiscriminate context expansion.

The choice of chunking strategy has a direct and often underappreciated impact on RAG system quality: chunks that are too small lack sufficient context for meaningful retrieval, chunks that are too large dilute the specific information relevant to a query, and chunks whose boundaries bisect a coherent argument or explanation can confuse both the encoder and the generator. We continue our discussion with a survey of three families of chunking strategies prominent in the literature today: layout-based chunking, fixed-window chunking, and semantic chunking.

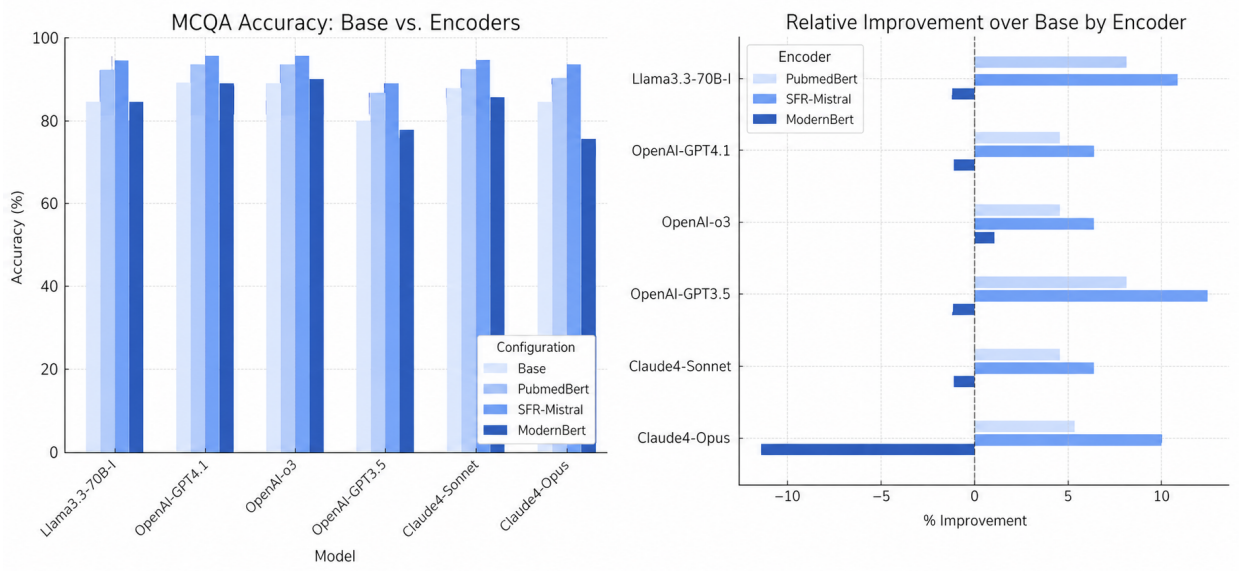

Figure 2.4: Effect of chunking granularity on MCQA performance. A misconfigured semantic chunking hyperparameter using ModernBERT embeddings [164] collapses entire documents into single chunks, causing retrieval to return a small set of full-length papers rather than targeted passages. When retrieving $K = 10$ such chunks (i.e., ten full papers), performance degrades consistently across all evaluated LLMs, highlighting the importance of properly calibrated chunking for effective retrieval-augmented reasoning.

2.5.1 Layout-Based Chunking

The most straightforward approach is to chunk text according to the structural boundaries created by the author and identified during parsing: paragraph breaks, section headers, page boundaries, or other formatting cues. Layout-based chunking has the advantage of aligning chunk boundaries with authors' own narrative structure. This strategy assumes that authors structure their writing with intention, i.e., a paragraph typically develops a single idea, a section covers a coherent topic, and a subsection addresses a specific aspect of that topic.

However, layout-based chunking suffers from two problems. First, the granularity of structural elements varies enormously across documents and within a single document: a methods section may contain a single dense paragraph spanning two pages, while a results section may consist of many short paragraphs interspersed with figure references. This variability produces chunks of wildly inconsistent length, complicating downstream encoding

(which naively truncates chunks that exceed its context length limits) and retrieval (where longer chunks dilute relevance signals)(Figure 2.4). Second, layout-based chunking is only possible when the parser preserves structural information, which extraction-based parsers often do not, and which even neural parsers may detect imperfectly.

2.5.2 Fixed-Window Chunking

Fixed-window chunking segments text into chunks of predetermined length, typically measured in characters, words, or tokens. A common configuration uses sliding windows of 500–1000 characters with 50–100 characters of overlap between consecutive chunks. The overlap ensures that concepts spanning a boundary are captured in at least one chunk.

Fixed-window chunking is simple, deterministic, and produces chunks of consistent length, properties that simplify downstream processing. It is the default strategy in many mainstream RAG frameworks, including early versions of LangChain [25] and LlamaIndex [93]. However, its fundamental weakness is that boundary conditions are completely decoupled from the semantic flow of the text. A fixed window may split a sentence in the middle, separate a claim from its supporting evidence, or isolate an equation from its surrounding explanation. These truncation artifacts degrade both encoding quality (the encoder produces a representation of an incoherent fragment) and retrieval quality (the fragment may match a query keyword without providing sufficient context to be useful).

Recursive character splitting, which is an improvement over fixed-window chunking, partially mitigates this by attempting to split at paragraph boundaries first, then sentence boundaries, then word boundaries, falling back to character boundaries only when the higher-level splits produce chunks that exceed the target length. This approach produces more coherent chunks than naive fixed-window splitting but remains fundamentally driven by length constraints rather than an awareness of the underlying semantic content.

2.5.3 Semantic Chunking

As of writing, semantic chunking is arguably the most sophisticated and context-aware text chunking algorithm in the literature. It places boundaries where the *meaning* of the text shifts, producing variable-length chunks that each encompass a coherent, self-contained line of thought. The core idea is to detect topic boundaries by measuring the semantic similarity between consecutive text segments (buffers): where similarity drops sharply, a topic shift has occurred, and a chunk boundary should be placed. Average chunk length is governed by a hyperparameter called *buffer size* which adjusts the number of sentences to be treated as an atomic unit. Chunks are constructed by merging buffers according to the following procedure. First, all buffers are embedded with a (preferably fast and lightweight) encoder. Then some mathematical measure of likeness such as cosine similarity or Euclidian distance between each buffer's embedding E_{B_i} and its consecutive neighbor $E_{B_{i+1}}$ is computed. This yields a distribution of distances (or similarities). Another hyperparameter adjusts for the percentile of this distribution that will be used as a cutoff, e.g., any pair of buffers that is more than 95% percentile apart from each other will mark a chunk boundary. An overview of the algorithm is depicted in Figure 2.5

To this day, the original inventor of the semantic chunking algorithm remains ambiguous. Some accounts attribute the contribution to Kamradt [77] who mentioned a variant of the logic during a tutorial video on YouTube in 2023, around roughly the same time when we experimented with a similar concept during a summer internship (unaware of this video). Regardless of the ambiguity in its origins, subsequent work has extended semantic chunking in several directions. LLM-based boundary detection proposed by Chen et al. [27] prompts a language model to identify topic shifts directly, capturing subtler transitions that might be diluted in pooled embeddings – albeit, at a substantially higher computational cost. Proposition-based chunking also introduced by the same authors decomposes text into atomic single-fact statements, producing maximally fine-grained chunks but again, at a substantial

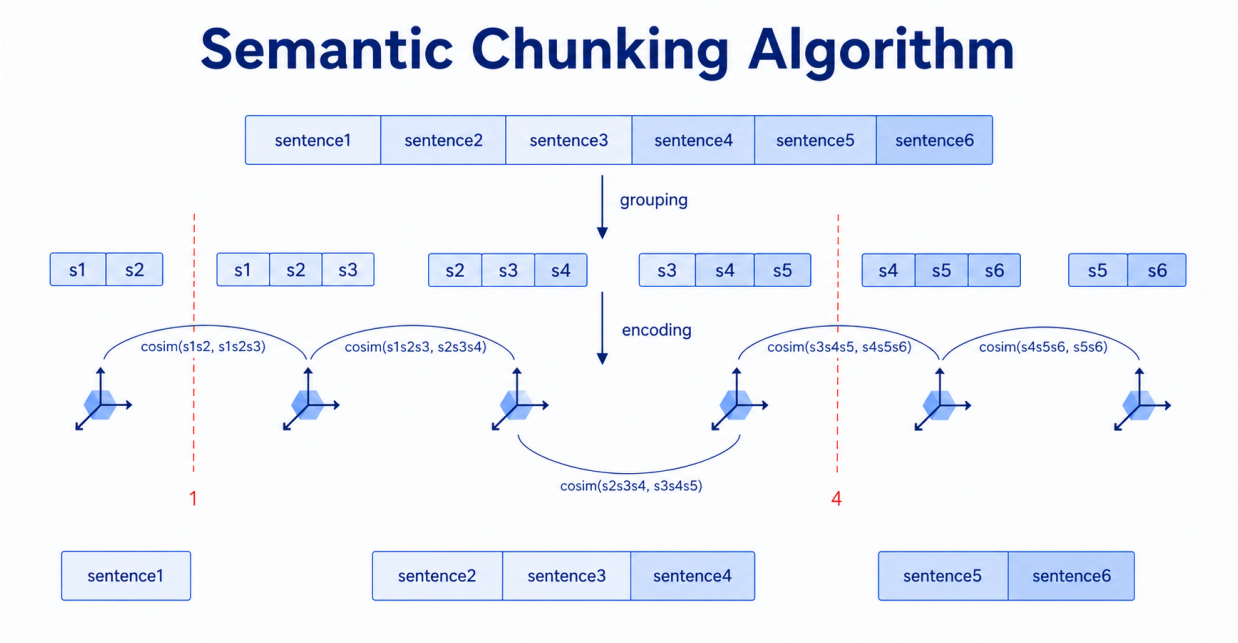

Figure 2.5: A visual depiction of the semantic chunking algorithm.

cost. Moreover, excessively fine-grained chunks lead to several downstream problems such as limited total context covered by top K retrieved chunks, storage requirements associated with increased number of embeddings in vector indexes, and myopic scopes that hinder generator's capability of synthesizing ideas across chunks.

The tradeoff between chunking quality and computational cost remains an active area of research. Chapter 3 presents the semantic chunking algorithm developed and deployed at supercomputer scale in HiPerRAG, which achieves content-adaptive boundaries at a cost only marginally higher than fixed-window chunking and scales to millions of documents on leadership-class supercomputers.

2.6 Neural Information Retrieval and Metric Learning

2.6.1 Dual Encoders and Cross-Encoders

The design space for neural retrieval models spans a spectrum from dual encoders (which independently encode queries and documents) to cross-encoders (which jointly encode query-document pairs).

Dual encoders, such as those we discussed in the context of Dense Passage Retrieval (DPR) [79], compute query and document representations independently, enabling efficient nearest neighbor search through precomputation and caching of document embeddings. The computational cost of retrieval is $O(d)$ per query-document comparison (where d is the embedding dimensionality) plus the cost of the ANN search itself. However, the independent encoding means that the document representation cannot adapt to the specific query, potentially missing fine-grained relevance signals.

Cross-encoders [117] jointly encode the concatenation of query and document through a Transformer, allowing full attention between query and document tokens. This produces much more accurate relevance scores but at prohibitive computational cost: every query-document pair requires a full forward pass through the model, making cross-encoders impractical for retrieval over large corpora. In practice, cross-encoders are used as rerankers: a dual encoder retrieves a candidate set of K documents (typically $K = 100$–1000), and the cross-encoder reranks these candidates.

The trade-off between the efficiency of dual encoders and the accuracy of cross-encoders motivates intermediate architectures, of which ColBERT's [80] late interaction mechanism (discussed in Section 2.6.4) is arguably the most influential.

2.6.2 Navigating the Scale vs Specificity Trade-off in Scientific Text Encoders

The choice of encoder is one of the most defining decisions for the overall performance of scientific RAG, and the design space presents a fundamental trade-off between the scale of general-purpose models and the domain specificity of smaller, specialized models.

Domain-specific encoders. The earliest approach to improving retrieval on scientific text was to pretrain encoder models on domain-specific corpora. SciBERT [17] (110M parameters), pretrained on 1.14 million papers from Semantic Scholar, established that domain adaptation yields consistent improvements on scientific NLP tasks. PubMedBERT [56] (110M parameters) went further by pretraining from scratch exclusively on PubMed text, demonstrating that domain-pure pretraining outperforms continued pretraining from a general-domain checkpoint. SPECTER [30] (110M parameters) adapted SciBERT for document-level similarity by training on citation-based triplets: if paper A cites paper B but not paper C, then A and B should be closer in embedding space than A and C. This citation-aware training produced embeddings that capture the relational structure of scientific literature, yielding strong performance on tasks such as recommendation, classification, and search. SPECTER2 [143] extended this approach with adapters for different tasks, recognizing that a single embedding space cannot simultaneously optimize for all downstream uses.

In the era of trillion-token, trillion-parameter frontier models, domain-specific encoders can be considered small—even tiny— at 110M parameters and 768-dimensional embeddings, trained on corpora that are comprehensive within their niche but minuscule relative to web-scale data. Nevertheless, they compensate for their lack of breadth through the depth of their highly focused training, which exposes them to even the most obscure intricacies of the underlying science captured in the outliers of their token distribution. Despite potential shortcomings in cross-disciplinary associations or global scientific patterns, they can perform remarkably well when chosen with a specific domain application in mind.

General-purpose encoders. The alternative is to use general-purpose text embedding models trained on diverse, large-scale data. SFR-Embedding-Mistral [134], a model that we widely study in this dissertation, fine-tunes the E5-Mistral-7B backbone with transfer learning across retrieval, clustering, and classification tasks, achieving strong zero-shot performance across domains including science. NV-Embed-v2 [86], also with 7B parameters, improves upon this by removing the causal attention mask, introducing a latent attention pooling layer, and applying a two-stage contrastive instruction-tuning recipe.

The Massive Text Embedding Benchmark (MTEB) leaderboard-the premier evaluation platform for encoder performance across model sizes, modalities, domains, and tasks [110]-attracts a vibrant research community of representation learning researchers. As of early 2026, proprietary multimodal models and larger open-weight encoders are in the lead, enforcing the de-facto established lessons that scale and instruction tuning transfer effectively to retrieval, and fine-tuning on domain-specific pairs yields further gains beyond what scale alone provides. The latter observation directly motivates the ColTrast fine-tuning approach introduced in Chapter 3.

The tradeoff in practice. In our experiments (Chapter 3), we find that domain-specific encoders such as PubMedBERT [56] outperform general-purpose encoders of similar size on domain-specific retrieval benchmarks, but that large general-purpose encoders (SFR-Embedding-Mistral) can close the gap through scale alone. The ColTrast fine-tuning algorithm introduced in this dissertation addresses this tradeoff directly: by fine-tuning a large encoder backbone (SFR-Embedding-Mistral with its 7B parameters) on scientific question-passage pairs with a combined contrastive and late-interaction objective, ColTrast produces an encoder that inherits the scale advantages of general-purpose models while acquiring the domain specificity and query-awareness needed for scientific retrieval.

2.6.3 Contrastive Learning for Text Encoders

The training of text encoders for retrieval relies on metric learning, which is the task of learning a distance function over a representation space such that semantically similar items are projected close together and dissimilar items are projected far apart. The dominant approach is contrastive learning, which trains the encoder to discriminate between positive (relevant) and negative (irrelevant) query-document pairs.

The InfoNCE loss [118], originally proposed for self-supervised representation learning, has become the standard objective for contrastive learning:

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(\mathbf{h}_q, \mathbf{h}_{d^+})/\tau)}{\exp(\text{sim}(\mathbf{h}_q, \mathbf{h}_{d^+})/\tau) + \sum_{i=1}^{N} \exp(\text{sim}(\mathbf{h}_q, \mathbf{h}_{d_i^-})/\tau)} \tag{2.5}$$

where $\mathbf{h}_q$, $\mathbf{h}_{d^+}$, and $\mathbf{h}_{d_i^-}$ are the encoder representations of the query, positive document, and i-th negative document, respectively; $\text{sim}(\cdot, \cdot)$ denotes cosine similarity; and τ is a temperature parameter that controls the sharpness of the contrastive distribution.

The temperature parameter τ plays a critical role in training dynamics. Small τ values sharpen the distribution, concentrating gradient mass on the hardest negatives and enabling fine discrimination but risking training instability. Large τ values flatten the distribution, promoting stable training but weakening the discriminative signal. The choice of τ is not cosmetic but has measurable impact on downstream retrieval quality [161].

The quality of contrastive learning also depends critically on the method of negative sampling (selection of negative examples). Random negatives are easy to discriminate and provide noisy and weak gradients. It is hard negatives, i.e., passages that suggest relevance through their semantic similarity to the query but are not actually relevant, who provide stronger learning signals. Xiong et al. [171] demonstrated that mining hard negatives via approximate nearest-neighbor search during training substantially improves retrieval quality compared to static or random negative sampling. In the same vein, Sentence-BERT[131]

applied contrastive learning to produce sentence embeddings suitable for semantic similarity tasks, establishing the popular 'sentence transformer' paradigm that underlies much of modern text retrieval. Subsequent work, including E5 [162] and GTE [90], has scaled sentence transformers to larger backbone models and more diverse training data, pushing the boundaries of embedding quality for general-purpose retrieval.

2.6.4 Late Interaction: ColBERT

While dual-encoder models compress each query and document into a single vector, this early pooling discards fine-grained token-level information that may be essential for determining relevance. Khattab and Zaharia [80] proposed ColBERT, which retains per-token embeddings for both queries and documents and computes relevance via a MaxSim operation:

$$S(q,d) = \sum_{i=1}^{|q|} \max_{j=1}^{|d|} \mathbf{E}_{q_i}^{\top} \mathbf{E}_{d_j} \tag{2.6}$$

where $\mathbf{E}_{q_i}$ and $\mathbf{E}_{d_j}$ are the contextualized embeddings of the i-th query token and j-th document token, respectively. For each query token, the maximum similarity to any document token is computed, and these maxima are summed to produce the final relevance score. This "late interaction" mechanism preserves token-level alignment, capturing, for instance, that the query token "BRCA1" should align specifically with mentions of BRCA1 in the document rather than being diluted into a single-vector representation of the entire passage.

ColBERT achieves a favorable precision–efficiency tradeoff between full cross-encoder reranking (subsection 2.6.1) and dual-encoder retrieval. Document token embeddings can be precomputed and stored, so the per-query cost is proportional to the number of query tokens times the number of candidate document tokens (in the reranking stage) rather than the size of the entire corpus. ColBERTv2 [136] refined the approach with residual compression (reducing the storage overhead of per-token embeddings) and denoised supervision (using a

cross-encoder to generate training targets).

The ColTrast objective introduced in this dissertation (Chapter 3) synthesizes the strengths of contrastive learning and late interaction. By jointly optimizing an InfoNCE contrastive loss on pooled embeddings and a MaxSim late-interaction loss on per-token embeddings, ColTrast trains encoders whose pooled representations benefit from the fine-grained alignment enforced by the MaxSim objective during training. At inference time, only the pooled embeddings are used, preserving the efficiency of standard dense retrieval while achieving representational quality that approaches late-interaction models.

2.7 Evaluation of LLMs and RAG Systems

2.7.1 Scientific QA Benchmarks

The evaluation of language models on scientific tasks has relied on a succession of benchmarks whose difficulty and specificity increased in parallel to the sophistication and capabilities of frontier LLMs.

As one of the earliest scientific NLP benchmarks which predates the concept of an LLM, SciQ [168] provides 13,679 crowdsourced science exam questions spanning physics, chemistry, and biology, each with their supporting passages. While SciQ is useful for evaluating general scientific knowledge, its questions are typically at the undergraduate level and do not call for the deep domain expertise necessary for cutting-edge scientific research.

PubMedQA [74] targets biomedical question answering with 1,000 expert-annotated questions derived from PubMed abstracts. Each question asks whether a yes/no/maybe answer is supported by a given abstract, testing the model's ability to interpret biomedical evidence.

MMLU [62] (Massive Multitask Language Understanding) encompasses 57 subject areas at varying difficulty levels, including several science subjects (anatomy, astronomy, biology, chemistry, etc.). MMLU has become one of the most widely used benchmarks for evaluating

general LLM capability, though its breadth comes at the cost of depth in any single domain. Moreover, there is strong evidence that, at this point, most of its content has leaked into the pretraining data of contemporary LLMs [88].

In the scientific literature domain, LitQA [84] provides expert-annotated questions that require information from full-text articles, testing whether RAG systems can extract and reason over evidence buried in scientific papers. ScholarQABench [4] extends this to multi-paper question answering, requiring synthesis across multiple sources.

At the edge of the difficulty spectrum reside benchmarks that prioritize innate reasoning and problem-solving capabilities over searching, retrieval, and interpretation skills. As one such example, GPQA [132] (Graduate-Level Google-Proof Q&A) targets expert-level questions in biology, physics, and chemistry that are specifically designed to be unsolvable via a Google web search by non-experts, requiring genuine domain understanding rather than information retrieval. A harder subset, GPQA Diamond [132], filters for only the most consistently difficult questions (defined as those those answered correctly by fewer than one-third of domain experts who are not the question authors) serving as a higher-stakes probe of genuine scientific reasoning at the frontier of human competence.

Humanity's Last Exam [124] (HLE) pushes this further still, crowd-sourcing 2,500 questions across mathematics, the natural sciences, and the humanities, each verified to be unambiguous, precisely graded, and resistant to internet retrieval. Developed by a globally distributed consortium of subject-matter experts, HLE was explicitly designed as a benchmark that current models cannot saturate: at the time of release, state-of-the-art models achieved well under 10% accuracy, compared to approximately 90% for human domain experts. As of today, the highest performing LLMs are proprietary closed-source models and they remain under 45%.

Most recently, ATLAS [92] (AGI-Oriented Testbed for Logical Application in Science) introduces a large-scale, contamination-resistant evaluation suite of approximately 800 origi-

nal problems spanning seven scientific disciplines—mathematics, physics, chemistry, biology, computer science, earth science, and materials science—all authored or substantially adapted by PhD-level domain experts. Unlike multiple-choice benchmarks, ATLAS requires open-ended answers with multi-step derivations in LaTeX, making it substantially harder to game and more faithful to the demands of real scientific inquiry.

2.7.2 The Benchmark Contamination Crisis

The reliability of existing benchmarks has come under increasing scrutiny due to benchmark contamination wherein the questions and answers in the benchmark leak into pretraining corpora through the internet. As benchmarks are published and shared online, their items become part of the web text that is subsequently crawled to create training datasets, artificially inflating performance estimates and undermining the real-life validity of evaluation.

Li et al. [88] documented striking contamination rates across major benchmarks: C-Eval (45.8% contaminated), MMLU (29.1%), ARC-Challenge (28.7%), and HellaSwag (12.4%). These rates have increased over time as benchmark items propagate online and are absorbed into successive generations of training corpora.

Deng et al. [37] provided behavioral evidence of contamination through a creative experimental design: they removed one answer option from multiple-choice questions and asked models to guess which option was removed. We find it necessary to open up a parenthesis here in order to emphasize the impact of the finding. The removed option was not necessarily the correct option which a well-trained frontier model can predict even if it was not included in the options. Rather, correctly predicting a missing wrong option implies that the model has memorized that wrong option since it can not be expected to correctly guess the wrong answer among countless possibilities. Therefore, the authors conclude that the model has seen the question in its contaminated pretraining data. On MMLU, GPT-4 achieved 57% accuracy and ChatGPT achieved 52% accuracy on this prediction task, both figures are far

above the 25% random chance baseline expected if the model had not memorized the specific answer set. This result is consistent with pretraining exposure to the exact benchmark items.

We posit that the findings of Li et al. and Deng et al. discussed above are expected manifestations of the fundamental problem motivating this dissertation. As we have already argued, the rate of scientific output has long surpassed any individual researcher's capacity to ingest and process new information, and the rapid, widespread adoption of LLMs as research assistants will only widen this gap. In these circumstances, continuing to rely on human expertise for scientific benchmark generation is nothing short of pitting humans against machines: low-throughput, expert-curated benchmarks will inevitably be outpaced by increasingly data-hungry models, making benchmark contamination not an anomaly to be corrected, but a structural inevitability to be designed around.

These findings have profound implications for the interpretation of LLM performance claims and motivate the development of automated benchmark generation pipelines that can produce domain-specific, provenance-tracked evaluation sets at scale. Such pipelines would enable the scientific community to generate custom benchmarks that are guaranteed to be unseen by any model at the time of evaluation. Chapter 4 presents our contribution in this direction.

2.8 High-Performance Computing for AI

The computational demands of modern AI workloads have increasingly intersected with the capabilities of leadership-class supercomputers. This section surveys the HPC systems and software infrastructure relevant to this dissertation.

2.8.1 Leadership-Class Systems

The Argonne Leadership Computing Facility (ALCF) operates several systems relevant to this work. Polaris, a 44-petaflop system, is equipped with 560 nodes, each containing four NVIDIA A100 GPUs (2,240 GPUs total), providing a testbed for GPU-accelerated AI workloads. Sunspot, when we conducted this research, served as a testbed for the then-upcoming Aurora exascale system, equipped with Intel Data Center GPU Max 1550 ("Ponte Vecchio") accelerators. Aurora, which came online in 2024, delivers over 2 exaflops of peak AI performance using over 60,000 Intel GPU Max series accelerators across 10,624 nodes, making it one of the most powerful systems in the world for AI workloads.

The Oak Ridge Leadership Computing Facility (OLCF) operates Frontier, the first system to break the exascale barrier, equipped with AMD MI250X accelerators. Frontier provides approximately 1.7 exaflops of peak double-precision performance and over 6 exaflops of mixed-precision AI performance across 9,408 nodes.

2.8.2 Workflow Orchestration

Parsl [11] (Parallel Scripting Library), developed at the University of Chicago (Globus Labs) and Argonne National Laboratory, provides the workflow orchestration layer used throughout this dissertation. Parsl is a pure-Python library that enables the expression of complex dataflow patterns through Python function decorators and their execution across heterogeneous computing resources. It abstracts away the tedious details of job scheduling, data

staging, resource provisioning, and fault tolerance, allowing researchers to express their computational workflows in familiar Python syntax while achieving efficient execution on leadership-class systems.

For the workloads in this dissertation, Parsl manages the distribution of PDF parsing tasks across GPU nodes (with warm-starting of neural models to avoid repeated model loading), the parallelization of semantic chunking across CPU nodes, the distributed training of ColTrast encoders across multi-GPU nodes, and the parallel execution of encoding and indexing operations. The ability to express all of these diverse workload patterns within a single workflow framework has been essential for the scaling experiments we present.

2.8.3 Distributed Training

Training large models across multiple GPUs requires careful orchestration of computation and communication. Data parallelism [89] replicates the model on each GPU and distributes the training data across replicas, synchronizing gradients after each forward-backward pass. Model parallelism [138] distributes the model's parameters across GPUs, enabling the training of models that exceed the memory capacity of a single device. Pipeline parallelism [112] combines elements of both by partitioning the model into stages and overlapping computation and communication across micro-batches.

For the ColTrast encoder training in this dissertation, we employ data parallelism with QLoRA [38] (Quantized Low-Rank Adaptation), which quantizes the pretrained model weights to 4-bit precision and trains low-rank adapter matrices in full precision. This approach reduces the memory footprint of fine-tuning a 7B-parameter model from approximately 28 GB to approximately 6 GB per GPU, enabling training on the A100 and Intel Max series GPUs available on ALCF systems. The distributed contrastive loss computation, which gathers embeddings across all workers to maximize the effective batch size, is implemented using PyTorch's distributed communication primitives.

CHAPTER 3

HIPERRAG: HIGH-PERFORMANCE RETRIEVAL-AUGMENTED GENERATION FOR SCIENTIFIC INSIGHTS

This chapter presents HiPerRAG, a distributed high-performance computing workflow for retrieval-augmented generation over scientific literature at the scale of millions of documents. The material in this chapter is based on Gokdemir et al. [52], published at the Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '25). In the following sections, we expand upon the published work with additional detail on design decisions, algorithmic specifics, and the translational applications of the system.

3.1 Introduction and Motivation

The challenges outlined in Chapter 1, i.e., parsing scientific documents, optimizing retrieval accuracy, evaluating RAG on scientific literature, and scaling to exascale infrastructure, collectively motivated the development of HiPerRAG. At its core, HiPerRAG embodies a simple proposition: that high-performance computing infrastructure, combined with algorithmic innovations in document parsing and neural information retrieval, can yield RAG systems that outperform both domain-specific models and state-of-the-art commercial LLMs on scientific question-answering tasks, and do so without requiring any fine-tuning of the generator model.

The proposition is significant because it inverts the conventional wisdom about how to improve LLM performance on scientific tasks. The dominant paradigm in the field has been to build ever-larger models or to fine-tune existing models on domain-specific data. Both approaches are expensive, both risk catastrophic forgetting, and both produce models whose knowledge is static at the point of training. HiPerRAG demonstrates an alternative path: keep the generator model frozen and general-purpose, and invest instead in the quality of

the retrieval infrastructure that equips the model with relevant evidence at inference time.

HiPerRAG indexes more than 3.6 million scientific articles sourced from bioRxiv, arXiv, and select journals. The system scales every component of the RAG pipeline, i.e., parsing, chunking, encoding, and retrieval, to thousands of accelerators across three leadership-class supercomputing platforms: Polaris at the Argonne Leadership Computing Facility (ALCF), Sunspot at ALCF, and Frontier at the Oak Ridge Leadership Computing Facility (OLCF).

The remainder of this chapter is organized as follows. Section 3.2 presents the detailed design of HiPerRAG's components: the Oreo document parser, the semantic chunking algorithm, the ColTrast encoder fine-tuning objective, the indexing and retrieval infrastructure, and the warmstart optimization that enables efficient distributed execution. Section 3.3 introduces the science-specific evaluation benchmarks developed for this work. Section 3.4 presents experimental results on document parsing quality, encoder retrieval accuracy, and end-to-end scientific question-answering performance. Section 3.5 details the scaling experiments on the three supercomputers. Section 3.6 surveys downstream applications of HiPerRAG in agentic scientific discovery workflows. Section 3.7 discusses the implications and limitations of the work.

3.2 System Design

Figure 3.1 depicts the high-level HiPerRAG workflow. A scientific corpus in PDF format enters the pipeline through Oreo, which extracts structured text from each document. The extracted text is segmented into semantically coherent chunks, which are encoded into dense vector representations by the ColTrast-finetuned encoder. These embeddings are indexed in a Faiss vector database. At query time, the user's question is encoded with the same encoder, nearest-neighbor chunks are retrieved, and the retrieved chunks are presented to a generator LLM alongside the original query to produce a grounded response. We now describe each component in detail.

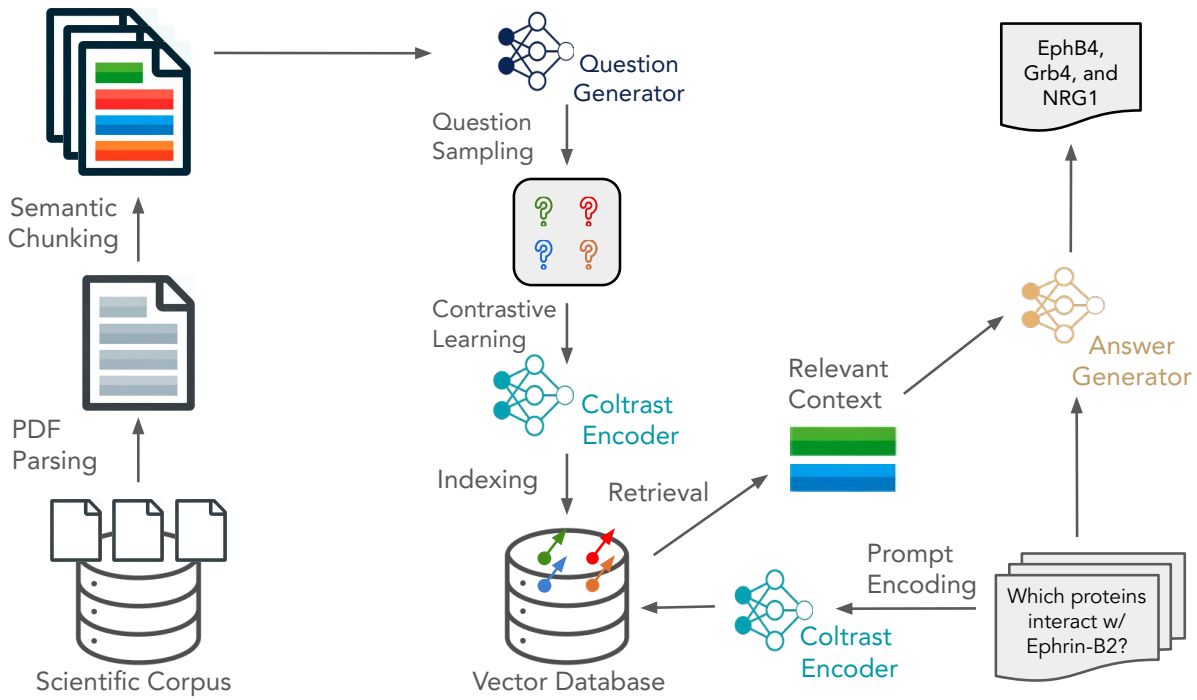

Figure 3.1: The HiPerRAG workflow. We implement a novel PDF parsing approach, namely Optical Recognition with Eclectic Output (Oreo), that takes into account the intrinsic formatting of multi-layout scientific manuscripts. We then design a query-aware encoder fine-tuned on scientific literature that semantically organizes the data into relevant chunks, which can be retrieved on a per-query basis. Both these innovations enable us to achieve computational efficiency and retrieval performance at scale.

3.2.1 Oreo: Optical Recognition with Eclectic Output

The first stage of the HiPerRAG pipeline is document parsing: converting raw PDF files of scientific articles into structured, machine-readable text. As discussed in Section 2.4, existing approaches to this task present a tradeoff between throughput and accuracy that none has resolved. Extraction-based parsers are fast but inaccurate and limited to text. End-to-end neural parsers are accurate and multimodal but computationally expensive. We introduce Oreo (Optical Recognition with Eclectic Output), a layout-aware multimodal parser that achieves the throughput of extraction-based methods together with the accuracy and multimodal coverage of neural approaches.

Architecture

Oreo operates in two stages, decomposing the parsing problem into layout detection and content recognition. Figure 3.2 depicts the overall workflow.

Stage 1: Layout Detection. Each page of a PDF is rendered as an image and passed through a YOLOv5 [130, 156] object detection model trained to identify regions of different content types. The model detects 20 categories of document elements, including text paragraphs, section headers, tables, table captions, figures, figure captions, mathematical equations, code blocks, references, and page metadata such as headers, footers, and page numbers. This stands in contrast to existing parsers: Marker [33] supports 3 categories (text, tables, equations), while Nougat [18] does not perform explicit layout detection and instead processes the entire page as a single image. Oreo's fine-grained layout detection enables category-specific processing strategies for each region and allows the pipeline to handle the diverse formatting conventions of different publishers and journals.

The choice of YOLOv5, a convolutional neural network, over a Vision Transformer for layout detection is deliberate. CNNs excel at the localization task of identifying bounding boxes of document elements while being substantially faster than ViTs of comparable accuracy. Layout detection does not require the long-range attention patterns that make ViTs powerful for image understanding; it requires fast, accurate bounding-box regression, which is the native strength of the YOLO architecture.

Stage 2: Content Recognition. Once layout detection has identified the bounding boxes of individual elements, the content of each text region is extracted using Texify, a Vision Transformer framework for image-to-text conversion. By restricting the ViT's attention to individual regions rather than entire pages, Oreo achieves a substantial reduction in computational cost. The quadratic scaling of self-attention ($O(n^2)$) means that processing a full page image (e.g., 1024×1024 patches) is far more expensive than processing k smaller

Figure 3.2: Parsing workflow for scientific PDFs. Oreo allows content-aware extraction of information from large-scale PDF collections by combining a YOLO-based layout detector with the Texify Vision Transformer for content-specific extraction. The two-stage design produces a 4.5× speedup over Nougat [18] and a 94.6× improvement in FLOP utilization, while supporting 20 categories of document elements.

regions of average size n/k. Specifically, the total cost of processing k regions of size n/k is $k \cdot O((n/k)^2) = O(n^2/k)$, which is k times cheaper than processing the full page.

This compartmentalized design produces a 4.5× speedup over Nougat and a 94.6× improvement in FLOP utilization, while supporting 20 categories of document elements compared to Nougat's full-page approach. Oreo is implemented in Python with PyTorch as the deep learning backend. The YOLOv5 layout detection model is trained on a dataset of annotated scientific document pages from diverse publishers, and the Texify content recognition model operates on cropped region images, producing Markdown output with LaTeX notation for mathematical content.

3.2.2 Semantic Chunking

Raw parsed text must be segmented into self-contained chunks suitable for retrieval. The choice of chunking strategy has a direct impact on retrieval quality. Chunks that are too small may lack sufficient context for meaningful retrieval, while chunks that are too large may dilute the specific information relevant to a query. We adopt the semantic chunking algorithm introduced in Section 2.5.3, which produces variable-length chunks by detecting topic boundaries in the text. The algorithm uses SFR-Embedding-Mistral [134] to encode overlapping sentence buffers, computes cosine similarities between consecutive buffer em-

beddings, and introduces a chunk boundary wherever consecutive embeddings fall below the K-th percentile of all pairwise similarities. We use $K = 5$ throughout this work, i.e., a chunk boundary occurs only where consecutive buffers are maximally dissimilar, indicating a topic shift.

Algorithm 1 Semantic Chunking

Require: Sentences $S = \{s_1, s_2, \ldots, s_n\}$, buffer size w, percentile threshold p
Ensure: Set of semantic chunks $\mathcal{C}$

// **Step 1: Construct buffers**
for $i \leftarrow 1$ to n **do**
    $B_i \leftarrow \bigcup_{j=\max(1,\, i-w)}^{\min(n,\, i+w)} s_j$
end for
// **Step 2: Embed buffers**
for $i \leftarrow 1$ to n **do**
    $\mathbf{e}_{B_i} \leftarrow \textsc{Encode}(B_i)$
end for
// **Step 3: Compute pairwise cosine distances**
for $i \leftarrow 1$ to $n - 1$ **do**
    $d_i \leftarrow 1 - \dfrac{\mathbf{e}_{B_i} \cdot \mathbf{e}_{B_{i+1}}}{\|\mathbf{e}_{B_i}\| \, \|\mathbf{e}_{B_{i+1}}\|}$
end for
// **Step 4: Determine boundary threshold**
$\tau \leftarrow \textsc{Percentile}(\{d_1, d_2, \ldots, d_{n-1}\}, p)$
// **Step 5: Place boundaries and build chunks**
$\mathcal{C} \leftarrow \{\}, \quad c_{\text{cur}} \leftarrow s_1$
for $i \leftarrow 1$ to $n - 1$ **do**
    if $d_i > \tau$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_{\text{cur}}\}$
        $c_{\text{cur}} \leftarrow s_{i+1}$
    else
        $c_{\text{cur}} \leftarrow c_{\text{cur}} \cup s_{i+1}$
    end if
end for
$\mathcal{C} \leftarrow \mathcal{C} \cup \{c_{\text{cur}}\}$ ▷ Append final chunk
return $\mathcal{C}$

This algorithm has several desirable properties. First, it is content-adaptive: chunk boundaries are determined by the semantic structure of the text rather than by arbitrary character or token counts. Second, it is parameterized by a single threshold (K), which

controls the granularity of chunking. Lower values of K produce fewer, larger chunks, while higher values produce more, smaller chunks. Third, it is embarrassingly parallel: each document's text can be chunked independently, enabling linear scaling across compute nodes.

Applied to the 3.6 million documents in the HiPerRAG corpus, the semantic chunking algorithm produces over 16 million chunks.

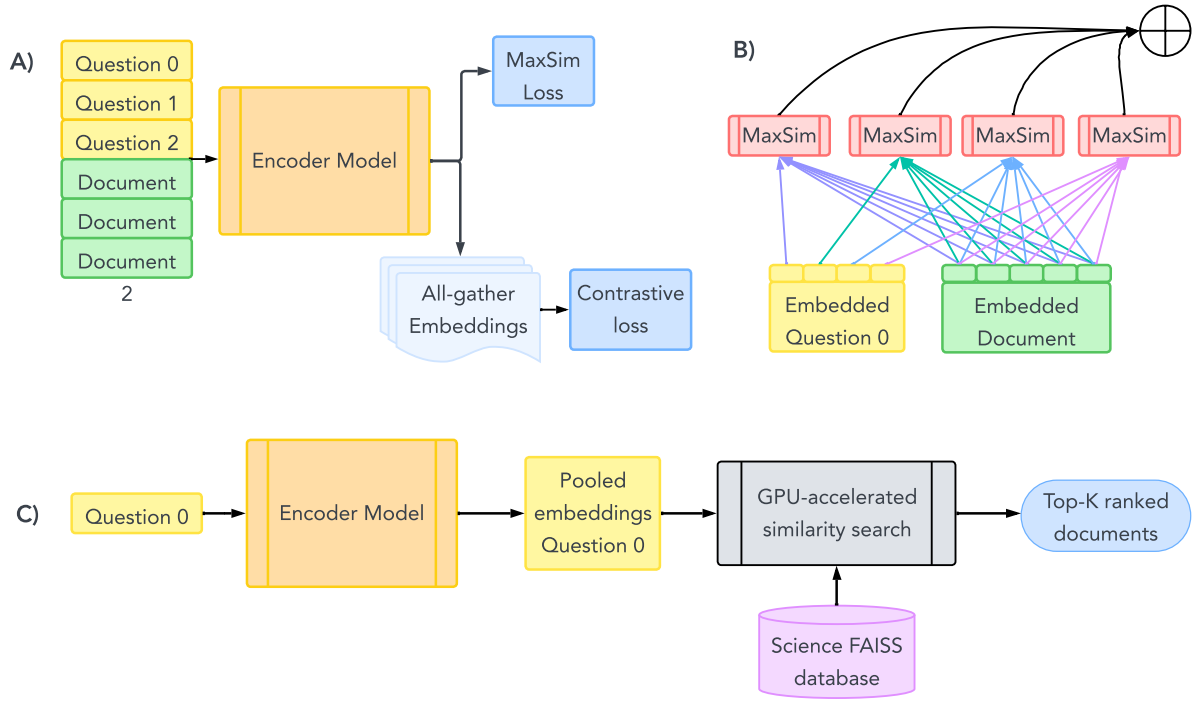

Figure 3.3: Encoder/retrieval workflow using the ColTrast loss method. (A) Training using ColTrast combines a local token-level late-interaction loss with a contrastive loss operating on gathered embeddings. (B) Details of the late-interaction loss, where each token in the question/query is compared to each document. (C) Inference-time workflow. Each question is embedded and used in similarity search to extract relevant chunks from the scientific database.

3.2.3 ColTrast: Query-Aware Encoder Fine-Tuning

The retrieval accuracy of a RAG system depends critically on the quality of the encoder's vector representations. As discussed in Section 2.6.2, general-purpose encoders trained predominantly on web text produce embeddings that may not capture the semantic nuances of scientific content. We develop ColTrast (Contrastive + Late-Interaction Training), a novel

fine-tuning algorithm that combines contrastive learning with late-interaction techniques to produce embedding representations tailored to scientific queries.

Design Decisions

The major determinants of retrieval performance are the similarity metric and the embedding quality, which is in turn determined by model size, loss function, and training data. We note that many of these performance determinants are double-edged: cutting back computation and improving scaling often comes at the cost of retrieval accuracy and vice versa. We identified several places where we could improve accuracy with no or minimal loss of scaling performance, as described below.

Similarity metric: token-by-token vs. pooled embeddings. Although ColBERT (Section 2.6.4) is a performant retriever in terms of retrieval accuracy, its late-interaction loss involves a token-by-token comparison that scales as the product of the numbers of tokens in the document chunk and query. For a large-scale vector store with billions of chunks of text, these scaling barriers become impractical. We therefore replace the token-by-token similarity metric with cosine similarity on the pooled embeddings at inference time. Normally, this would lead to performance degradation. We describe in the following how we overcome this problem by enhancing the accuracy of the embedding model.

Embedding performance: model size. For a well-trained model, model size is often a primary determinant of model performance. While BERT-style models such as ColBERT perform well on embedding tasks, especially for their size ($\sim$110M parameters), billion-parameter GPT-style models have started to surpass moderate BERT-style models on the embedding leaderboard. At the time of writing, a 7B-parameter GPT-style model named SFR-Embedding-Mistral [134] has the best overall performance on embedding tasks. It is therefore likely that SFR-Embedding-Mistral is a good basis for enhancing the performance

of our similarity metric that uses pooled embeddings.

Embedding performance: loss function. Fine-tuning on relevant tasks can enhance performance. However, increasing the model size by 70× relative to BERT limits our batch sizes and requires model sharding across GPUs, both of which hurt our ability to fine-tune the embedding representation. To resolve these issues, we fine-tune the 8-bit version of SFR-Embedding-Mistral using QLoRA [38] with a batch size of 24 per GPU. Since there is no need for model sharding, we can use simple data parallel approaches (e.g., ZeRO-1) and achieve linear scaling up to 400 nodes (1,600 GPUs) of the Polaris supercomputer (see Section 3.5 for details).

Training Objective

We use a combination of contrastive loss and late-interaction loss to fine-tune our embedding models. Contrastive loss benefits from large batch sizes [12]. Naively increasing batch size by reducing loss from all ranks and applying contrastive or late-interaction (LI) loss is limited by dense embeddings becoming large with increased world size, an effect that affects LI more acutely than contrastive losses. We therefore apply LI loss to the local rank only due to its memory and computational demands. For the contrastive loss, pooled embeddings from all ranks are gathered, and loss is calculated with the local rank compared to $\min(N, W)$ samples, where N is the maximum to consider and W is the total samples across all ranks.

The total loss per iteration is

$$\mathcal{L} = \frac{\mathcal{L}_{\text{LI}} + \mathcal{L}_C}{2}, \tag{3.1}$$

where $\mathcal{L}_{\text{LI}}$ is the maxsim loss (Equation (2.6)) and $\mathcal{L}_C$ is the contrastive loss (Equation (2.5)).Figure 3.3 illustrates the resulting ColTrast training and inference workflow. Following this fine-tuning algorithm produces an embedding representation that clusters related scientific concepts (Section 3.4.2).

Training Data

We fine-tune ColTrast on three scientific paper datasets that we curated, totaling 82,750 papers. These papers span diverse topics, including low-dose radiation and cancer biology, antimicrobial peptide literature, and SARS-CoV-2 research. The high-level composition of the combined dataset is summarized in Table 3.1, and the sub-domain distribution of the antimicrobial peptide dataset is illustrated in Figure 3.4 for additional clarity on its diversity.

Figure 3.4: Sub-domain distribution of the antimicrobial peptide dataset for ColTrast fine-tuning. The corpus spans bioactivity, antibiotic resistance, peptide structure and modification, nanotechnology applications, immune-system interaction, and gene therapy, illustrating the breadth of biomedical sub-domains over which the encoder is fine-tuned.

To enhance the effectiveness of our retrieval system, we apply the semantic chunking algorithm of Section 3.2.2 to the text of these papers. Such organization of text before vectorization is widely recognized to improve the precision and relevance of retrieved documents. For each semantic chunk created, we then employ Mistral-7B-Instruct-v0.2 [70] to generate a high-level question that uses the chunk as a reference or resource for its answer.

Table 3.1: Composition of the ColTrast fine-tuning dataset. Three scientific paper collections covering peptides, cancer biology, and SARS-CoV-2 research, totaling 82,750 papers and 851,900 semantic chunks, are used to generate 455,894 question-chunk pairs for fine-tuning.

Domain	PDF Count	Chunk Count	Generated Questions
Peptides	10,124	123,685	70,866
Cancer	18,034	208,803	114,535
COVID-19	54,592	519,412	270,493
Total	82,750	851,900	455,894

This process results in a collection of 455,894 question-chunk pairs, collectively referred to as the ColTrast fine-tuning dataset (Table 3.1). This structured approach not only improves the efficiency of our retrieval system but also significantly enhances the quality of generated questions, facilitating more accurate and contextually relevant responses.

3.2.4 Indexing and Retrieval

The encoded chunks are indexed using Faiss [41] for efficient approximate nearest-neighbor (ANN) search. We use an IVF-PQ (Inverted File with Product Quantization) index, which partitions the embedding space into Voronoi cells and compresses each embedding using product quantization. This index type provides a favorable tradeoff between search speed, memory consumption, and recall.

At query time, the user's question is encoded with the same ColTrast-finetuned encoder, and the top-k nearest neighbors ($k = 5$ by default) are retrieved via cosine similarity. The retrieved chunks, along with their source document metadata (title, authors, DOI), are presented to the generator LLM in a structured prompt template alongside the question. The generator LLM is Mixtral-8x7B-Instruct [71], a mixture-of-experts model that activates approximately 13B parameters per token while maintaining a total parameter count of 47B. We choose Mixtral for its favorable quality-to-cost ratio: it achieves performance competitive with much larger dense models at substantially lower inference cost due to sparse activation.

Table 3.2: Semantic chunking model load times on Polaris and Sunspot increase at larger scales. The 2- and 128-node values correspond to Polaris, while the 4- and 96-node values correspond to Sunspot. All times are in seconds.

	Nodes				
System	2 / 4	32	64	96 / 128	256
Polaris	36 ± 21	88 ± 28	114 ± 24	130 ± 23	362 ± 88
Sunspot	172 ± 73	218 ± 107	209 ± 160	535 ± 5	—

3.2.5 Distributing HiPerRAG with HPC and Warmstart Optimization

We use the Parsl parallel programming library [11] to distribute the execution of our PDF parsing, semantic chunking, chunk encoding, and question sampling workflows. Parsl supports execution across diverse HPC resources and schedulers and has been shown to scale to hundreds of thousands of workers and thousands of tasks per second. However, standard Parsl assumes that tasks are pure functions, which is not conducive to persisting shared data structures, such as ML models, between tasks. For example, loading models for semantic chunking is an I/O-heavy operation and takes longer when more nodes read the same weight files concurrently: nearly nine minutes at worst, as shown in Table 3.2.

We therefore implement a model registry that makes Parsl workers stateful actors that can persist a shared state across task invocations. The registry is implemented as a module-level global singleton variable that caches the return result of a Python function or class initialization in a dictionary using a hash of the arguments and keyword arguments as a unique key. The registry permits only one object at a time. Before a new object is created and registered, any existing object is automatically destroyed to free up shared resources, such as GPU memory [22]. In our use case, the registry captures a reference to the model upon its first initialization within a worker process (cold start), then returns the cached model for all subsequent task invocations (warm start). This method enhances system utilization by minimizing I/O overheads, amortizing the cost of model loading, and keeping the model in device memory (e.g., CUDA memory) to reduce memory transfers from host to device.

3.3 Science-Specific Evaluation Benchmarks

Question-answering (QA) datasets are pivotal for both training and evaluating LLMs in various scientific domains. These benchmarks are especially valuable for gauging a model's depth of domain-specific knowledge, its ability to understand intricate questions, and its proficiency in quantifying uncertainty during the generation of open-ended responses. As discussed in Section 2.7, existing scientific QA benchmarks are valuable but limited in scope and domain coverage. We introduce three new evaluation resources designed to test HiPer-RAG on domain-specific biomedical tasks.

ProteinInteractionQA (7591 Q/A pairs). We compiled a dataset of 16,009 antimicrobial peptides from three sources: the Antimicrobial Peptide Database (APD), the Database of Antimicrobial Activity and Structure of Peptides (DBAASP), and the Database of Antimicrobial Resistance Peptides (DRAMP). For each peptide, we used the UniProt API to retrieve data on proteins that interact with them. This effort identified 7,591 peptides with known interactants. Using the instruction-tuned Mistral7B model [70], we generated multiple-choice questions from this experimentally validated data. The resulting task poses questions such as "What protein does <*peptide-name*> interact with?" The correct answer includes all known interactants from UniProt, while the distractors comprise non-interactants randomly selected from other peptides in the dataset.

ProteinFunctionQA (17646 Q/A pairs). To curate ProteinFunctionQA, we downloaded functional descriptions for 17,646 antimicrobial peptides from the UniProt database. We then employed the Mistral7B instruction-tuned LLM [70] to generate multiple-choice questions based on this experimentally annotated data. The correct answer reflected the ground-truth function of the peptide as retrieved from UniProt, while the distractors were incorrect functions sampled from other peptides within the dataset. The task requires selecting the correct function for a given peptide.

BioSynthQP (1500 Q/P pairs). We created this synthetic biomedical dataset by using GPT-4 prompt engineering with the express purpose of evaluating retrieval accuracy. BioSynthQP features scientific questions surrounding subdomains of medicine such as virology, oncology, and cardiology. For each subdomain, we generate a set of questions, along with 10 paragraphs for each question with a decreasing level of accuracy and relevance. The most relevant paragraph constitutes an answer that contains keywords, empirical results, and citations. The least relevant paragraph, on the other hand, digresses away from the question, lacks scientific basis and citations, but still is broadly related to medicine. We apply human supervision to the postprocessing of the synthetic content to guarantee the accuracy of the quality labels assigned to each sample. Consequently, BioSynthQP presents a demanding retrieval task that evaluates an encoder's capability to generate embeddings that accurately capture quality characteristics within samples from the same domain.

3.4 Experimental Results

It is important to reiterate that all RAG systems are made up of multiple independent components. Accordingly, HiPerRAG consists of the Oreo document parser, the ColTrast query-aware encoder, and an arbitrary generator model to produce a response in light of the relevant content retrieved. Our discussion in this section commences with an evaluation of Oreo in Section 3.4.1. The evaluation of ColTrast in retrieval accuracy follows in Section 3.4.2. Section 3.4.3 concludes with an evaluation of various HiPerRAG configurations on five scientific Q/A benchmarks, two of which are our contribution.

3.4.1 Parser Evaluation

We evaluate Oreo, Marker, and Nougat on $n = 100$ scientific documents spanning eight domains (mathematics, physics, chemistry, biology, engineering, medicine, economics, and computer science) and six publishers (ArXiv, BioRxiv, MedRxiv, BMC, MDPI, and Nature).

Table 3.3: Accuracy and throughput of image-based parsers on diverse scientific documents.

	Oreo (ours)	Nougat	Marker
Accuracy (BLEU) [%]	46.34	46.42	56.90
Accuracy (CAR) [%]	73.92	70.92	73.51
Throughput [PDFs/GPU sec]	0.55	0.12	0.09

Ground-truth text is sourced from the full-text HTML versions of the articles. Parsing quality is measured using bilingual evaluation understudy (BLEU) [123] and character accuracy rate (CAR). BLEU evaluates n-gram overlap between parsed and ground-truth text. CAR measures character-level precision, particularly valuable for scientific equations and numerical data.

The results in Table 3.3 show that Marker achieves the highest BLEU score but has the lowest throughput. Nougat offers a competitive BLEU score but the lowest CAR. Notably, Oreo achieves the highest CAR and throughput while approximately matching Nougat's BLEU performance. Therefore, Oreo maximizes the number of accurately parsed tokens, making it a suitable option for large-scale document parsing.

In addition to the across-the-board accuracy and throughput, Oreo also surpasses both Nougat and Marker in multimodal capabilities. Specifically, Oreo can distinguish between 20 types of document assets, compared to Marker's three. Unlike Nougat, which can parse only text from PDFs, Oreo comprehensively handles a variety of document assets including figures, tables, equations, and code. Furthermore, Oreo explicitly parses metadata, which enables the storage of citations, author names, and references from scientific documents to facilitate digital archiving.

3.4.2 Encoder Evaluation

To evaluate the ColTrast retrieval approach, we use two datasets: (1) our BioSynthQP dataset described in Section 3.3, and (2) a 5% held-out evaluation dataset from the ColTrast

Table 3.4: Model performance on the BioSynthQP dataset and evaluation split of the data. Abbreviations used include: BSQP – BioSynthQP dataset; Eval – the evaluation split; M – Salesforce Research/SFR-Mistral base model; B – BERT base model; Q – quantized low-rank approximation (QLoRA) parameter-efficient fine-tuning; ND – No distributed all-gather in contrastive loss; CL – contrastive loss only; LI – Late-Interaction loss only; S – model with ColTrast loss (CL + LI).

Name	P@10 BSQP	MRR@10 BSQP	MRR@20 Eval	Top-20 Eval
ColTrast-M-Q-S	0.74	0.35	0.93	0.62
ColTrast-M-Q-ND-S	0.81	0.51	0.90	0.98
ColTrast-B-S	0.33	0.03	0.21	0.05
ColTrast-B-ND-S	0.72	0.46	0.84	0.96
BERT-CL-S	0.28	0.45	0.184	0.04
BERT-LI-S	0.03	0.003	0.06	0.003
PubMedBERT	0.72	0.65	0.22	0.39
ColBERT-v2	0.31	0.11	0.371	0.97
BERT-Base	0.31	0.03	0.21	0.37

training set described in Section 3.2.3. Rather than sampling the evaluation set randomly, we ensure that the training and evaluation sets contain passage/query pairs from distinct scientific documents. Table 3.4 shows results of training with different model architectures and loss paradigms against the evaluation datasets. When provided with a question, the documents are ranked according to the cosine similarity between the question and document embeddings.

We report the precision as $P = N_{rel}/N_{ret}$ for the number of relevant retrieved documents, N_{rel}, and the total number of retrieved documents, N_{ret}. In the case of P@10, N_{ret} is set to 10. MRR@10 is calculated as $MRR = N_q^{-1} \sum_i \text{rank}_i^{-1}$ where rank_i is the ranking of the positive paired document for the i^{th} query, N_q is the total number of queries, and @10 sets that if the positive document is ranked beyond 10, that term of MRR is zero.

All models in the upper section of Table 3.4 were trained with the IA^3 [**?**] parameter-efficient fine-tuning (PEFT) method, except ColTrast-M-Q-S, since the QLoRA approach requires LoRA PEFT [66]. We performed ablations to examine the effectiveness of the

ColTrast combined loss method. BERT-CL-S uses contrastive loss only, BERT-LI-S only has late-interaction loss, and <*Model name*>-ND-S denotes models with the ColTrast loss method but excluding the all-gather for the contrastive loss. The base model used for fine tuning is denoted by "M" for SFR-Mistral and "B" for BERT-base. We observe that performance on evaluation metrics is improved with the ColTrast loss method, as opposed to utilizing only LI or contrastive losses, and that, for these models where a decent batch size is allowed, ND performs comparably to the distributed loss. The models in the lower half of the table are pre-existing approaches to the embedding and retrieval task included for comparison.

3.4.3 End-to-End Scientific Question Answering

We analyze the performance of our scientific Q/A from three angles: composition of the retrieval corpus, encoder model employed for retrieval, and the generator that leverages the retrieved content to answer questions. The two scientific corpora that we curated for this analysis differ greatly in terms of their domain composition and document size. As shown in Table 3.1, the Protein Literature Corpus (PLC) consists of 10,124 PubMed articles about proteins. We encode this corpus with PubMedBERT, an encoder-based LLM specifically fine-tuned on PubMed articles [56]. On the other hand, the Scientific Literature Corpus (SLC) contains over 3.6 million articles across numerous domains. We encode SLC with our ColTrast-B-S and ColTrast-M-Q-S encoders, which have been fine-tuned on a comprehensive scientific dataset (Section 3.4.2). We use instruct-finetuned versions of two generators, Mistral-7B [70] and Mixtral-8x7B [71].

Table 3.5 compares encoder/generator combinations with respect to accuracy on five scientific Q/A tasks. SciQ [168] covers crowd-sourced science exam questions on Biology, Chemistry, and Physics. PubMedQA [74] contains biomedical research questions. LitQA [84] also focuses on the biomedical domain, but features questions that can only be answered from

Table 3.5: Comparison of different encoder models with the Protein Literature Corpus (PLC) and the Scientific Literature Corpus (SLC). Results are reported as accuracy (%) on multiple-choice scientific question-answering benchmarks. Abbreviations used include: PMB – PubMedBERT encoder; ColTrast-B-S – ColTrast encoder based on BERT-base; ColTrast-M-Q-S – ColTrast encoder based on SFR-Mistral with QLoRA. PIQA and PFQA refer to ProteinInteractionQA and ProteinFunctionQA, respectively.

Corpus	Generator (Encoder)	SciQ	PubMedQA	LitQA	PIQA	PFQA
PLC	Mistral (PMB)	0.80	0.45	0.32	0.44	0.32
	Mixtral8x7B (PMB)	0.88	0.76	0.34	0.70	0.52
SLC	Mistral (ColTrast-B-S)	0.82	0.45	0.32	0.40	0.32
	Mistral (ColTrast-M-Q-S)	0.85	0.44	0.24	0.43	0.332
	Mixtral8x7B (ColTrast-B-S)	0.90	0.75	0.34	0.66	0.526
BASE	Mistral	0.88	0.59	0.40	0.44	0.36
	Mixtral8x7B	0.78	0.73	0.38	0.70	0.36

knowledge in full-text papers. ProteinInteractionQA and ProteinFunctionQA are presented in this work (Section 3.3). We also provide baseline results where we do not use retrieval.

We observe that the Mixtral8x7B model that retrieves with PubMedBERT outperforms GPT-4 (75.2% accuracy) on the PubMedQA benchmark despite having access to a much smaller biomedical corpus (PLC) and having a smaller parameter count (14 billion parameters utilized at inference). Additionally, this model also outperforms PubMedGPT [20], a domain-specific LLM with 2.7 billion parameters pre-trained from scratch on biomedical data. This result suggests that retrieval-augmented generation at scale can improve the accuracy of LLMs in a data-efficient manner and render general-purpose LLMs more performant than domain-specific models on scientific QA tasks. Moreover, the Mixtral8x7B model that uses the ColTrast-B-S encoder answers 90% of the questions in the comprehensive SciQ dataset. ColTrast-B-S retrieval and the SLC corpus lead to a 12% improvement of this model on that benchmark. This result is consistent with our hypotheses that combining a novel metric learning-based retrieval strategy with the ability to retrieve from millions of scientific articles can equip LLMs with domain-specific knowledge beyond their training data.

3.5 Scaling Experiments

We conduct extensive scaling experiments on the document parsing, semantic chunking, and encoder fine-tuning steps of HiPerRAG. We evaluate performance and scale on three diverse supercomputing systems: Polaris at the Argonne Leadership Computing Facility (ALCF), Sunspot at ALCF, and Frontier at the Oak Ridge Leadership Computing Facility (OLCF). The architectural details and node-level specifications of these systems were discussed in Section 2.8.1. Figure 3.5 summarizes the strong-scaling results across all three pipeline stages.

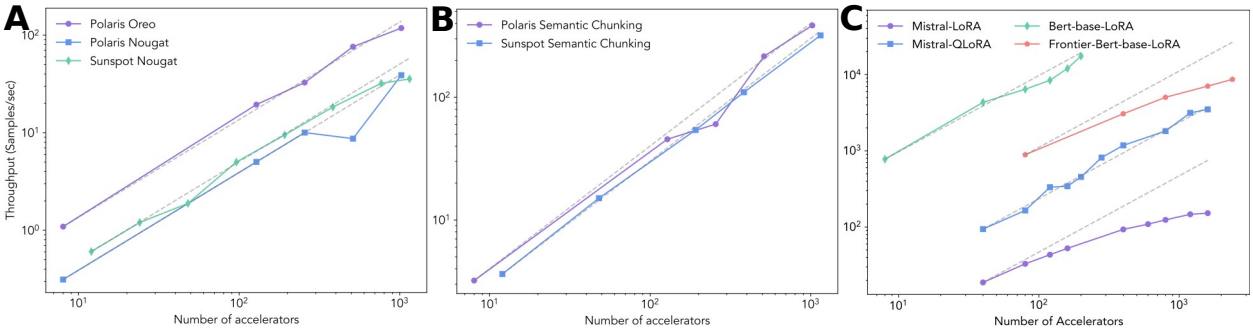

Figure 3.5: Strong scaling of HiPerRAG components. (A) PDF parsing throughput for Oreo and Nougat on Polaris and Sunspot. (B) Semantic chunking throughput on Polaris and Sunspot. (C) Encoder fine-tuning throughput across BERT-base, Mistral-LoRA, and Mistral-QLoRA configurations on Polaris, with the BERT-base configuration also run on Frontier. Unless noted, a run was accomplished on Polaris. Note that one "GPU" on Frontier in panel C corresponds to one Graphics Compute Die (GCD).

3.5.1 PDF Parsing

Figure 3.5A shows the performance of the Oreo and Nougat parsers as we strong-scale on Polaris and Sunspot by increasing the number of accelerators to 1024 GPUs on Polaris (256 nodes) and to 1152 tiles on Sunspot (96 nodes, each with 6 GPUs and 12 tiles). On Polaris at 1024 accelerators, Oreo achieves 118.2 samples/s and Nougat 38.9 samples/s, a 3× improvement in throughput with Oreo over Nougat.

We also compare the performance of Nougat processing on Polaris and Sunspot. We

observe a throughput of 35.62 samples/s on 1152 tiles (96 nodes) of Sunspot and 38.9 samples/s on 1024 GPUs (256 nodes) of Polaris. At a node-to-node level, we observe a 2.36× improvement in performance on Sunspot over Polaris. This result is expected, as we have more GPUs and memory on each Sunspot node. At an accelerator level, comparing a single A100 GPU to an Intel Max GPU tile, we observe a 26% improvement for an A100 GPU in comparison to a tile on the Intel Max Data Center GPU. We attribute this primarily to the current state of optimization of the software stack here. In all cases, we observe linear scaling in throughput as we scale out, which we attribute to the embarrassingly parallel nature of the parsing workflows.

For end-to-end parsing, Oreo achieves 40.7 TFLOPS/GPU peak with a sustained average of 37.2 TFLOPS/GPU on an NVIDIA A100, while Nougat achieves 0.43 TFLOPS/GPU peak and a sustained average of 0.28 TFLOPS/GPU. Thus, Oreo achieves 94.6× better compute performance than Nougat while sustaining a 4× improvement in throughput. This result is expected, as Oreo follows a compartmentalization strategy that adds compute requirements in two ways. First, layout detection and combination of text items require tensor operations. Second, once text items are transcribed, they need to be mapped to their proper position. These innovations enable Oreo to efficiently process multiple file formats and multi-modal document assets including figures, tables, equations, and code.

3.5.2 Semantic Chunking

Figure 3.5B shows the strong-scaling performance of our end-to-end semantic chunking workflow on Polaris and Sunspot. We observe linear scaling in terms of throughput in samples/s as we scale with the number of accelerators. This result is expected given the embarrassingly parallel nature of the workflow. We observe a slight dip in performance at 256 accelerators (64 nodes), likely due to storage system limitations when reading the data to be chunked. We observe a throughput of 385.7 samples/s on 1024 accelerators (256 nodes) of Polaris.

On Sunspot, we observe a throughput of 319.8 samples/s on 1152 accelerators (96 nodes). Normalizing these at an accelerator level, we observe a 35% improvement in achievable throughput on an A100 GPU in comparison to a tile of an Intel GPU. In terms of a node-to-node comparison, we achieve a 2.2× improvement on a Sunspot node in comparison to a Polaris node.

3.5.3 Encoder Fine-Tuning

Figure 3.5C shows strong scaling for training encoder models with samples/s as our throughput metric. Polaris runs include the major model architectures used here: BERT-base and Mistral. Each model is trained using parameter-efficient fine-tuning, both for computational efficiency and to help the model retain its knowledge from pretraining. Early experiments showed less than ideal scaling for Mistral-LoRA. Since the model is too large for a single GPU, we sharded it across ranks by using DeepSpeed ZeRO-3, and further employed parameter and optimizer state offloading in order to maximize batch size. To overcome this poor scaling, we adopted QLoRA training, where the model is first quantized to reduce memory pressure and then a LoRA adapter is applied as trainable parameters in full (bf16) precision. The Mistral-QLoRA approach yielded twofold benefits: first, the reduced memory footprint allowed us to load the full Mistral model into a single NVIDIA A100 with a batch size of 24; second, the reduced communication overhead compared to ZeRO-3 also enabled ideal scaling up to 400 nodes of Polaris.

Despite the success of QLoRA on Polaris, we were unable to employ a similar tactic on Frontier, as the BitsAndBytes quantization library required by transformers is not yet supported on MI250X. Performance in terms of TFLOPS/GPU, total FLOPs, and sample throughput are enumerated for the scaling runs in Table 3.6. Notably, applying the QLoRA approach to SFR-Embedding-Mistral fine-tuning increases TFLOPS/GPU by 132×, and also achieves the highest TFLOPS/GPU and total FLOPs of all measured models. Further, given

Table 3.6: Estimated floating-point operations and throughput for embedding/retriever model architectures during 100-step scaling runs using 400 GPUs on the Polaris supercomputer (<*model*>-P). BERT-base-LoRA-F was run on 2400 GCDs (1200 GPUs) on the Frontier supercomputer. Throughput is in samples per second; total samples (S_T) in millions of samples.

Name	TFLOPS/ GPU	FLOPs (10^{18})	Samples/s	S_T (10^6)
Mistral-LoRA-P	0.71	2.40	152.2	0.64
Mistral-QLoRA-P	93.80	160.00	3496.4	3.84
BERT-base-LoRA-P	7.95	0.67	24,157.0	2.56
BERT-base-LoRA-F	1.92	3.10	8654.8	7.68

the total document chunks of the SLC dataset (16.4M samples), Mistral with QLoRA could iterate an epoch of data in 1.3 hours. Such throughput enables fine-tuning on corpora of hundreds of millions of PDF documents.

3.6 Applications of HiPerRAG

Beyond its performance on benchmarks, HiPerRAG has been deployed as the retrieval backbone for several downstream scientific systems. This section surveys two such applications, illustrating the versatility of HPC-scale RAG infrastructure when integrated into broader scientific workflows.

3.6.1 StructBioReasoner: Multi-Agent Biologics Design

The most comprehensive integration of HiPerRAG into an agentic framework is StructBioReasoner [141], a scalable multi-agent system for designing biologics targeting intrinsically disordered proteins (IDPs). IDPs comprise between 30% and 40% of the human proteome and approximately 80% of cancer-associated proteins contain long intrinsically disordered regions, making these proteins essential therapeutic targets. Yet IDPs have historically been classified as "undruggable" because they lack the stable three-dimensional structures and

well-defined binding pockets that conventional structure-based drug design methods require.

StructBioReasoner employs a tournament-based reasoning framework in which specialized agents iteratively generate, evaluate, and select among competing hypotheses spanning target hotspots, design strategies, and binder scaffolds. Figure 3.6 depicts the overall architecture. The system integrates a Planning and Reasoning agent that coordinates the design process, a Literature Explorer agent powered by HiPerRAG, an AI Structure Prediction agent, a Molecular Dynamics Simulation agent for conformational sampling, and a Free Energy Calculation agent for binding affinity estimation. These agents are orchestrated via Academy, an extensible federated agentic middleware that coordinates agent execution across DOE supercomputing resources.

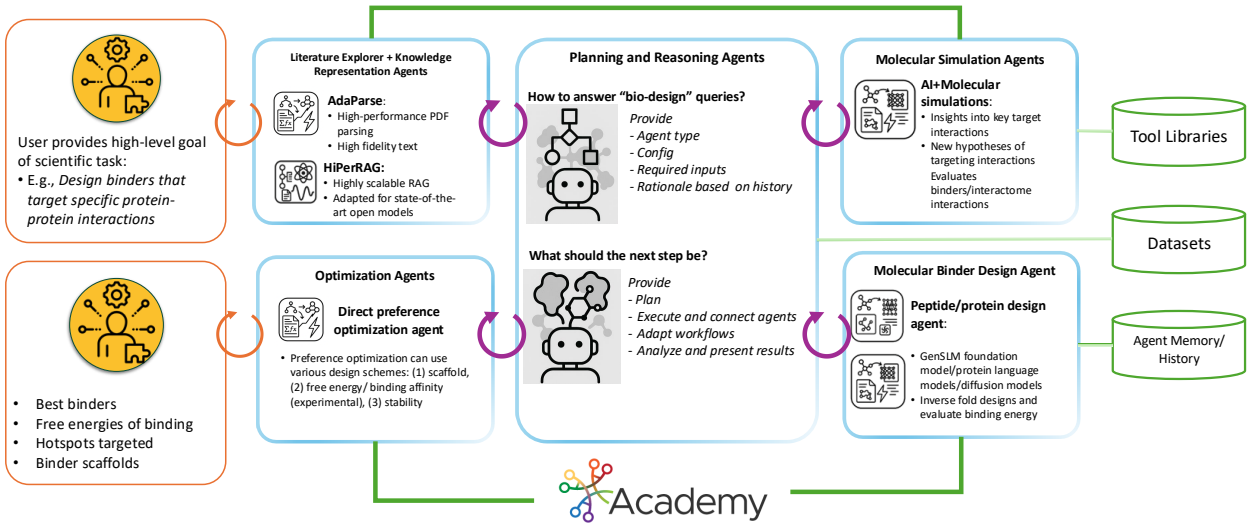

Figure 3.6: Overview of the StructBioReasoner multi-agent architecture for biologics design. The system coordinates specialized agents for literature review, hypothesis generation, structure prediction, molecular dynamics, and free energy calculation which are all orchestrated through the Academy [76] agentic framework. Central to the design loop is the integration of HiPerRAG as a dedicated agent responsible for efficient and scalable retrieval from scientific articles. The evidence surfaced by HiPerRAG directly informs hypothesis generation and binder design decisions made by the Reasoning agent, grounding the agentic workflow in peer-reviewed scientific knowledge. Figure adapted with consent from Sinclair et al. [141].

The HiPerRAG agent serves as the literature synthesis component. When the Reasoning agent formulates a hypothesis about potential protein-protein interactions or binding modes,

it queries HiPerRAG's vector store of scientific articles to retrieve evidence supporting or contradicting the hypothesis. For instance, in the NMNAT-2 case study, the HiPerRAG agent retrieved literature on the NMNAT-2:p53 interaction interface, providing the Reasoning agent with evidence that guided the selection of binding modes for further computational validation.

In benchmark experiments, StructBioReasoner produced strong results on two systems of increasing disorder. For the Der f 21 allergen, the framework achieved a 50.98% success rate in producing binders with more favorable binding free energy than the BindCraft-designed reference, out of 787 designs that passed quality control. For NMNAT-2, the system identified three distinct binding modes from 97,066 designed binders that passed sequence and structural validation, including the well-studied NMNAT-2:p53 interface. The discovery of this interface was initiated through HiPerRAG's retrieval of relevant protein-protein interaction evidence from the indexed literature, demonstrating that HPC-scale retrieval can contribute directly to scientific discovery when embedded within an agentic reasoning loop.

The StructBioReasoner deployment also demonstrated scaling on HPC infrastructure: the MD Simulation agent achieved 80.4% efficiency at 256 nodes, the Free Energy agent achieved 82.8% parallel efficiency at the maximum node count tested, and the Binder Design agent achieved 84.4% efficiency at 256 nodes.

3.6.2 BV-BRC CoPilot: RAG for Bioinformatics

HiPerRAG has been deployed as the retrieval engine powering the CoPilot feature of the Bacterial and Viral Bioinformatics Resource Center (BV-BRC) [139], one of the most widely used data platforms in microbial bioinformatics. BV-BRC serves as a comprehensive resource for bacterial and viral genomic data, hosting genome assemblies, annotations, protein structures, and comparative genomics tools for the global microbiology research community. As of August 2025, BV-BRC supports over 70,000 registered users who submit over 20,000

analysis jobs each month.

BV-BRC CoPilot is an AI-powered natural-language assistant embedded throughout the BV-BRC website to help users navigate data, services, and job results in real time. It is powered by a local instance of the Llama 4 Scout open-source LLM, combined with a retrieval-augmented generation layer built on HiPerRAG. Figure 3.7 shows the two ways in which CoPilot can be launched: from the chat icon at the bottom-right corner of any page, which passes the current page metadata as additional query context, or from the green action bar that appears after performing a search or launching an analysis, which passes selected data items or analysis job metadata as query context.

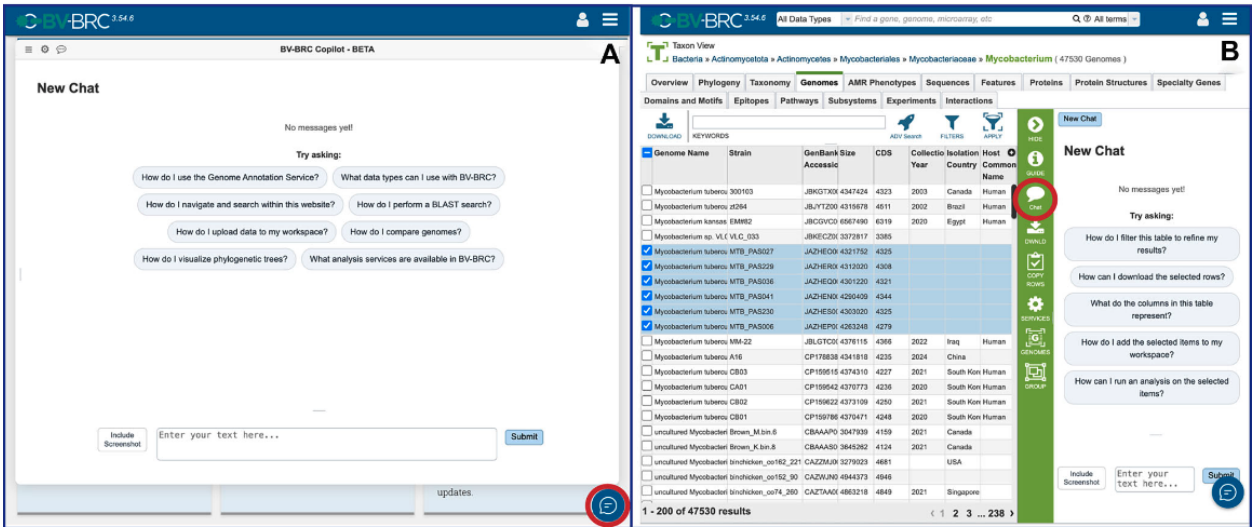

Figure 3.7: BV-BRC CoPilot, an AI-driven natural-language interface to the BV-BRC resource powered by HiPerRAG. CoPilot can be launched (A) using the chat icon at the bottom-right corner from any page, which passes the current web page as query context, or (B) using the chat icon from the green action bar, which passes selected data items or analysis job metadata as query context. Figure adapted with consent from Shukla et al. [139].

For this application, in addition to indexing BV-BRC tutorials, user guides, FAQs, and publications, HiPerRAG indexes approximately 440,000 papers from the American Society for Microbiology (ASM) journal collection. We are currently developing an expanded index featuring millions of documents across numerous venues. The CoPilot allows incorporation of relevant context and content from the active web pages, including screenshots, as additional

context for user queries to provide accurate and context-aware responses. The BV-BRC CoPilot represents one of the first deployments of supercomputer-scale RAG infrastructure in a production bioinformatics platform, demonstrating that the HPC-scale parsing, chunking, and indexing capabilities of HiPerRAG can be leveraged to create practical tools for the scientific community.

3.7 Discussion

HiPerRAG demonstrates that the co-design of retrieval algorithms and HPC infrastructure can yield scientific RAG systems that outperform both domain-specific models and frontier commercial LLMs on scientific question-answering tasks. The result carries several important implications.

Retrieval quality over model scale. The central finding, that a Mixtral8x7B mixture-of-experts model retrieving with PubMedBERT over the Protein Literature Corpus surpasses a GPT-4-class model operating without retrieval on PubMedQA, supports the thesis that retrieval quality is a primary determinant of downstream accuracy in knowledge-intensive scientific tasks. This finding suggests that investments in retrieval infrastructure can yield greater marginal returns than scaling model parameters alone.

Composability of RAG infrastructure. The deployment of HiPerRAG across multiple downstream applications, e.g., StructBioReasoner and BV-BRC CoPilot, demonstrates that well-designed RAG infrastructure is composable: it can serve as a reusable component within diverse scientific workflows without modification to the core retrieval system. This composability is a consequence of the modular design: the parsing, chunking, encoding, and retrieval stages are independent components connected by well-defined data interfaces (raw PDFs $\rightarrow$ parsed text $\rightarrow$ chunks $\rightarrow$ embeddings $\rightarrow$ Faiss index).

Limitations. We note four limitations of HiPerRAG. First, Oreo's layout detection relies on YOLOv5, which may fail on highly unconventional document layouts (e.g., poster-style presentations, multi-fold brochures). Second, ColTrast's effectiveness is bounded by the quality and diversity of the question-passage training data: if the training questions do not span the range of queries that scientists actually ask, the encoder may not generalize well to novel query types. Third, the system currently operates on text-only retrieval; scientific figures, tables, and equations contain information that is not fully captured by text extraction, even with Oreo's multimodal parsing. Fourth, the reliance on a frozen generator LLM means that HiPerRAG cannot benefit from improvements in the generator's reasoning capabilities unless the generator itself is upgraded.

Toward a computable scientific corpus. HiPerRAG's indexing of 3.6 million papers represents approximately 1.8% of the estimated 200 million scientific papers in existence. Scaling to the full corpus would require approximately 55× more compute for parsing and indexing, a target that is well within the capacity of current exascale systems. The Aurora supercomputer at ALCF, for instance, could complete this task in approximately two weeks using 5,000 nodes. This calculation suggests that the vision of a single, unified index over all of published science is not merely aspirational but technically feasible with current infrastructure.

CHAPTER 4

AUTOMATED BENCHMARK GENERATION AND REASONING-TRACE RETRIEVAL FOR DOMAIN ADAPTATION

This chapter presents a scalable pipeline for generating multiple-choice question-answering (MCQA) benchmarks from scientific literature and introduces reasoning-trace retrieval as a strategy for adapting small language models to scientific domains. The material is based on Gokdemir et al. [51], published at the SC '25 Workshop on Frontiers in Generative AI for HPC Science and Engineering, and is expanded here with additional analysis, methodological detail, and discussion.

4.1 Introduction and Motivation

The evaluation of language models on scientific tasks depends on benchmarks that are simultaneously domain-specific, uncontaminated by pretraining data, and reflective of current scientific knowledge. As discussed in Section 2.7.2, none of these requirements are consistently met by existing benchmarks. MMLU test items show 29.1% contamination with pretraining corpora [88], ARC-Challenge items are contaminated at a rate of 28.7%, and GPT-4 demonstrates behavioral evidence of memorizing specific answer sets from widely used benchmarks [37]. These findings undermine the validity of performance claims based on these benchmarks and call into question whether improvements on contaminated benchmarks reflect genuine advances in model capability or merely improved memorization.

The problem is structural rather than incidental. Static benchmarks – created once and published openly – inevitably propagate into the training data of subsequent models. The web-scale crawling that produces pretraining corpora cannot reliably exclude benchmark items, and even deliberate efforts to filter them are undermined by the items' redistribu-

tion across forums, study guides, and discussion threads. The half-life of a benchmark's uncontaminated status appears to be measured in months rather than years.

This structural problem motivates a shift from static to dynamic evaluation: rather than creating fixed benchmarks that ossify, we need pipelines that can generate fresh, domain-specific, provenance-tracked evaluation sets on demand. Such pipelines would enable the scientific community to generate benchmarks tailored to any subdomain of science at arbitrary scale, track the provenance of each question to its source document and chunk, regenerate benchmarks as the literature evolves so that evaluation keeps pace with the frontier of knowledge, and produce questions that are guaranteed to be unseen by any model at the time of evaluation, eliminating contamination by design.

This chapter presents our contribution toward this vision: a modular, HPC-scalable pipeline that transforms large corpora of scientific papers into quality-controlled MCQA benchmarks with strict provenance. As a case study, we apply the pipeline to radiation and cancer biology, generating 16,680 questions from 22,548 open-access articles and abstracts. We then make a second contribution – the discovery that reasoning traces distilled from frontier models serve as powerful retrieval sources for smaller models, enabling them to approach or exceed frontier-model performance on domain-specific tasks. To our knowledge, this represents the first systematic exploration of retrieval as a distillation mechanism for scientific domain adaptation.

The contributions of this chapter, summarized, are as follows. First, we present a scalable, modular pipeline for automated MCQA benchmark generation from scientific literature, designed to utilize high-performance computing platforms. Second, we release a new benchmark of 16,680 quality-controlled questions derived from 14,115 open-access papers and 8,433 abstracts in radiation and cancer biology. Third, we provide a systematic evaluation of small language models in the 1.1B–14B parameter range with retrieval-augmented generation from both paper-derived semantic chunks and frontier-model reasoning traces.

Fourth, we present empirical results showing that reasoning-trace retrieval consistently improves small models towards domain proficiency – often more so than directly retrieving from the underlying literature – and in some cases leads them to exceed the performance of GPT-4 on an expert-annotated domain examination.

4.2 The MCQA Generation Pipeline

The pipeline operates on a corpus of scientific papers and proceeds through five stages, each of which is designed to be modular, parallelizable, and scalable to HPC infrastructure via Parsl [11]. The pipeline builds on the parsing and chunking infrastructure developed for HiPerRAG (Chapter 3), extending it with question generation, quality control, and provenance tracking capabilities. Figure 4.1 provides an overview of the end-to-end workflow.

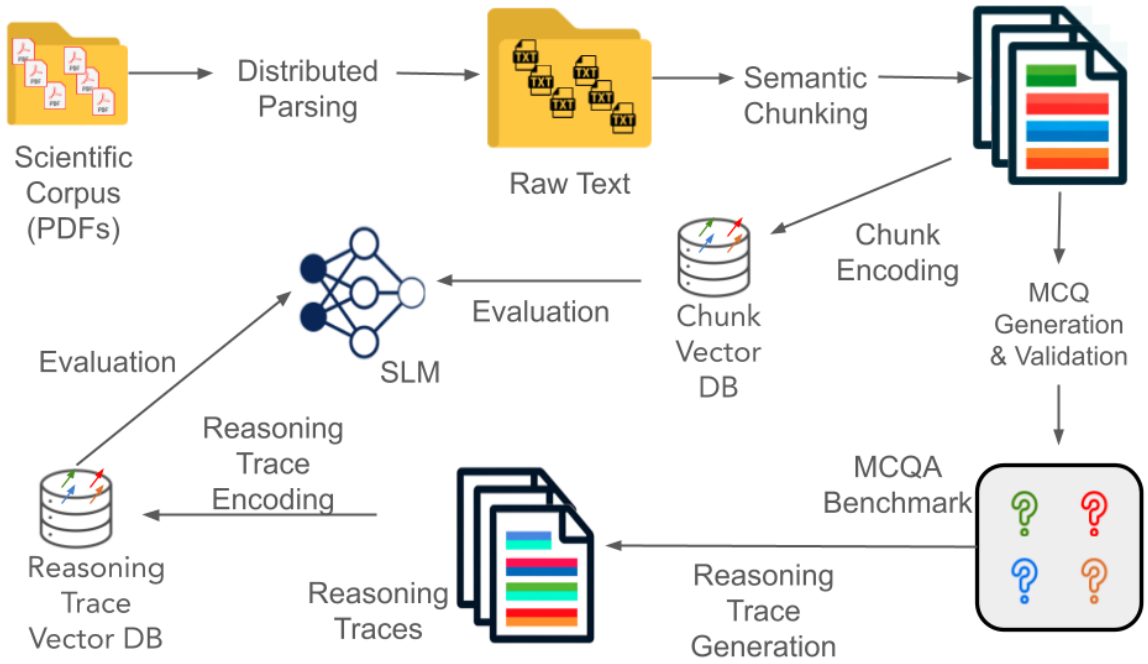

Figure 4.1: Workflow overview. PDFs are parsed into text and chunked semantically. Chunks are encoded with PubMedBERT for retrieval and also passed to GPT-4.1 for MCQ generation. Questions are then used for reasoning-trace generation and stored in a separate retrieval database. Models are evaluated under three conditions: i) no retrieval, ii) chunk retrieval, and iii) reasoning-trace retrieval. An LLM judge performs the grading and provides a justification.

4.2.1 Corpus Assembly and Parsing

For the case study in radiation and cancer biology, we assemble a corpus of 14,115 full-text open-access papers and 8,433 abstracts via the Semantic Scholar Academic Graph API [6], queried using cancer and radiation biology keywords. Full-text PDFs are parsed using the AdaParse [140] infrastructure, which assigns each document to an appropriate parser based on predicted accuracy and computational cost, producing JSON output with the extracted text and document metadata. For the abstract-only subset, text is extracted directly from the Semantic Scholar API metadata without the need for PDF parsing.

4.2.2 Semantic Chunking

Parsed text is segmented into semantically coherent chunks using PubMedBERT [56], a 330M-parameter encoder pretrained on biomedical text. We use PubMedBERT in lieu of the SFR-Embedding-Mistral [134] encoder used elsewhere in this dissertation (Chapter 3) because its biomedical pretraining makes it particularly well-suited for the radiation and cancer biology domain. The chunking algorithm follows the approach described in Section 3.2.2: overlapping sentence buffers are encoded, pairwise cosine similarities are computed between consecutive buffer embeddings, and a chunk boundary is introduced wherever consecutive embeddings fall below the chosen percentile threshold of all pairwise similarities. Applied to our corpus, the chunking stage produces 173,318 chunks. These chunks serve as the atomic units from which questions are generated – each question is derived from a single chunk, establishing a direct provenance link from question to source text.

4.2.3 Question Generation

For each chunk, a frontier LLM (GPT-4.1, accessed through Argonne's Argo-Proxy API [40]) generates a single multiple-choice question with seven answer options using a structured, two-stage prompting strategy.

The first stage produces a summary and expansion of the chunk's content. The LLM generates a 1–2 sentence summary identifying the central concept or finding, then expands on the identified concept by articulating its broader context and significance in the domain. The second stage produces the multiple-choice question itself: one correct answer and six plausible but incorrect distractors, with the prompt explicitly instructing the model to ensure exactly one unambiguously correct answer, to make the distractors plausible but clearly incorrect to a domain expert, to reference a concept that is directly mentioned in the source chunk, and to avoid references to outside content – and especially to avoid references to the source text by phrases like "according to the passage" or "as stated in the article" – so that the question is self-contained and can be answered without seeing the source chunk.

This two-stage strategy is superior to single-shot question generation because the summarization and expansion steps prime the model with a structured understanding of the chunk's content before it generates the question, reducing the likelihood of superficial or ambiguous questions.

4.2.4 Quality Control

Each candidate question undergoes automated quality assessment via a second-stage prompt to GPT-4.1. The quality assessment evaluates the question across four dimensions: clarity (is the question unambiguous and well-phrased?), correctness (is the designated correct answer actually correct?), distractor plausibility (are the distractors plausible enough to be challenging but clearly incorrect to a domain expert?), and educational value (does the question test a meaningful concept rather than trivial or surface-level information?).

Each dimension is scored on a scale of 1–10, and questions whose composite score falls below 7 are discarded. This filtering step is aggressive: of the 173,318 candidate questions, only 16,680 (approximately 9.6%) survive, yielding the final benchmark. The aggressive filtering reflects a deliberate preference for quality over quantity – a 16,680-question benchmark with

a high pass-rate threshold is more useful as an evaluation instrument than a 173,318-question benchmark that includes many marginal items.

4.2.5 Provenance Tracking

Each surviving question is stored in a structured JSON schema that records the question text and answer options, the correct answer, the source chunk text, the source document path, and metadata including timestamps, the cleaning version, the relevance check (score, type, reasoning), and the quality check (score, critique, raw output). Figure 4.2 shows the schema. This provenance chain – from question to chunk to paper – enables full traceability, supporting transparency, reproducibility, and the ability to regenerate or update the benchmark as the source corpus evolves.

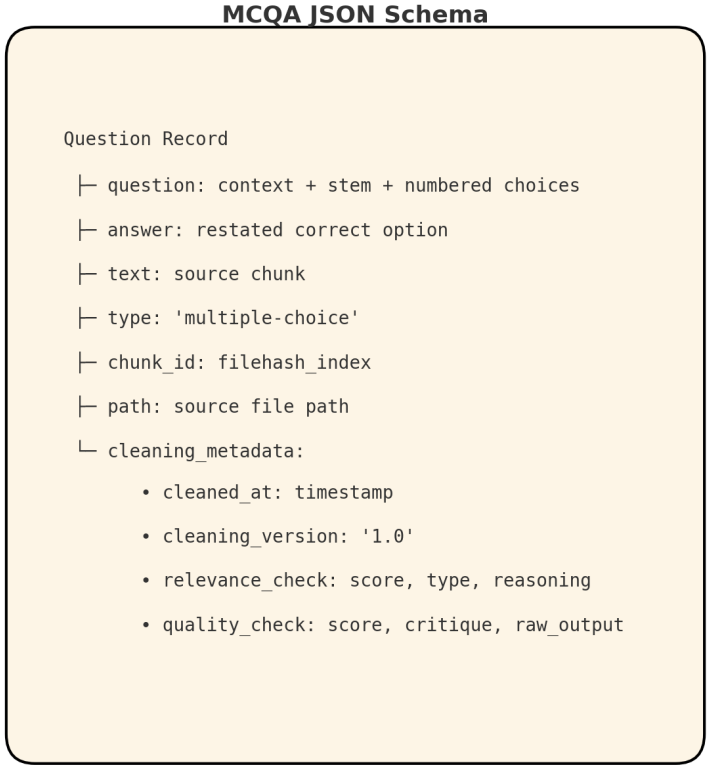

Figure 4.2: JSON schema for generated questions. Each record contains lineage to the source chunk and source PDF, along with relevance and quality checks to ensure transparent quality assurance.

4.3 Reasoning Traces as a New Modality of Knowledge Distillation

The second contribution of this chapter extends beyond benchmark generation to address a fundamental question: *how can the reasoning capabilities of frontier models be made accessible to smaller, deployable models without modifying the smaller models' weights?*

As discussed in Section 2.2.3, knowledge distillation has progressed from soft-target transfer [63] through representation distillation [73] to reasoning-trace distillation [34, 111]. Each advance transferred richer training signal – from logits, to hidden states, to explicit chains of thought – but all shared a common mechanism: the student model's weights were modified to internalize the teacher's knowledge. This weight-based approach has two limitations for scientific applications. First, distillation must be repeated for each domain, as a student distilled on biomedical reasoning will not generalize to materials science without additional training. Second, the distilled knowledge is frozen at training time and cannot reflect new literature without retraining.

We propose an alternative that we term *distillation through retrieval*: rather than using reasoning traces to train the student, we store them as retrievable artifacts in a vector index. At inference time, the student retrieves relevant traces and uses them as in-context scaffolding, gaining access to the teacher's reasoning process through the context window rather than through modified parameters. This mechanism is model-agnostic (any student can retrieve), domain-flexible (traces from different domains coexist in the same index), dynamic (traces can be updated as the teacher improves or the literature evolves), and transparent (traces are human-readable, enabling auditing and curation).

The unit of distillation in our framework is the *reasoning trace*: the chain of tokens that a reasoning LLM generates prior to producing its final answer. In models like OpenAI's o1 and GPT-4.1 with extended thinking, these traces can span hundreds or thousands of tokens and encode detailed step-by-step reasoning, i.e., identifying relevant principles, elim-

inating incorrect options, reconciling competing evidence, and building toward a justified conclusion. Reasoning traces are ordinarily ephemeral – generated during inference and discarded after the answer is extracted. Our approach treats them instead as durable, reusable computational artifacts.

Reasoning traces possess three properties that make them especially effective retrieval sources for domain adaptation. First, they are semantically dense and focused. Unlike raw scientific text, which may discuss many topics tangentially, a reasoning trace is organized entirely around answering a specific question. Empirically, the reasoning traces for our benchmark occupy approximately 50 MB of storage compared to 747 MB for the FP16 PubMedBERT embeddings of the raw semantic chunks from the same papers – roughly a 15× compression ratio that reflects the focused nature of traces. Second, they encode pedagogical structure: not just what the answer is but *how an expert would think through the question*. This thinking scaffold provides small models with a structured reasoning path that they can follow, rather than requiring them to reconstruct the reasoning from scratch based on raw evidence. Third, they enable cross-document synthesis. When the teacher model generates a reasoning trace, it draws on its full parametric knowledge – potentially encompassing information from thousands of papers that the student model has never seen. The trace captures this synthesized knowledge in a compact, retrievable form. The student model benefits from this synthesis without needing to perform it independently, and without requiring the underlying papers to be individually retrieved and processed.

The distinction between distillation through training and distillation through retrieval can be framed in terms of where the teacher's knowledge resides after transfer. In weight-based distillation, the knowledge is absorbed into the student's parameters: it becomes implicit, distributed across millions of weights, and inseparable from the student's other learned representations. In retrieval-based distillation, the knowledge remains external: it resides in the vector index as discrete, inspectable artifacts that the student accesses on

demand. This externality is what enables the model-agnosticism, domain-flexibility, and updatability that weight-based distillation lacks.

4.3.1 Trace Generation

For each MCQ in our benchmark, we prompt GPT-4.1 to produce three types of reasoning traces, generated simultaneously in a single API call and stored in separate Faiss [41] indices. These three traces correspond to three modes of reasoning at progressively increasing levels of abstraction. The detailed mode produces a comprehensive option-level analysis: a step-by-step trace that walks through the problem, considers each answer option, identifies relevant principles, and builds toward a justified conclusion. The focused mode produces a key-principle plus elimination trace: a more concise rationale that identifies the central principle needed to answer the question and briefly explains why each distractor is incorrect. The efficient mode produces a compact, high-level rationale that provides only the essential reasoning needed to discriminate the correct answer from the distractors.

Crucially, we explicitly instruct the teacher model to *omit the final answer* from each reasoning trace. This design choice ensures that the retrieved trace provides reasoning scaffolding without directly revealing the answer, testing whether the student model can use the reasoning to arrive at the correct answer independently. If the trace contained the answer, retrieval would trivially boost accuracy by converting the task from question answering to answer extraction. Figure 4.3 depicts the JSON schema for the resulting reasoning-trace records.

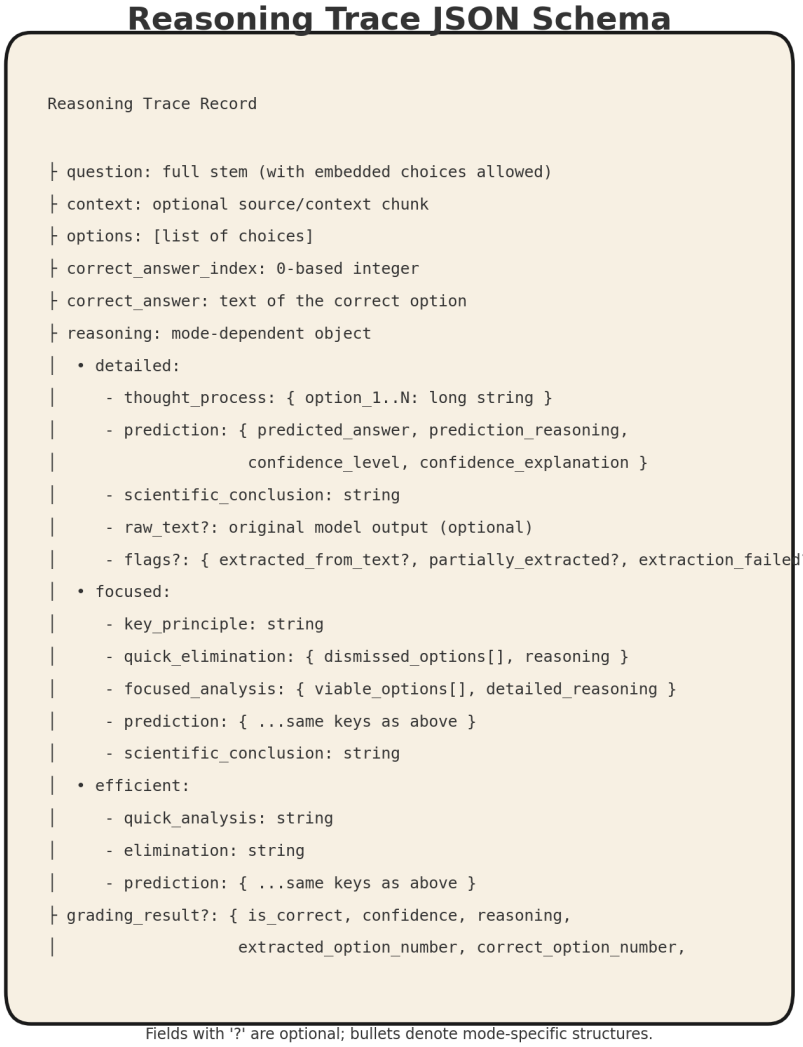

Figure 4.3: Reasoning-trace JSON schema. Supports three reasoning modes: detailed (option-level analysis), focused (principle + elimination), and efficient (compact high-level reasoning). Each record stores the question, context, options, correct answer index and text, the mode-dependent reasoning fields, and the grading result.

4.4 Experimental Setup

4.4.1 Model Selection

We evaluate a current and representative set of small and mid-sized open-source language models ranging from 1.1B to 14B parameters. The models were selected to capture the

current diversity of architectures, training corpora, and licensing terms available in the open-source community. This range of model sizes was chosen for three reasons. First, our case study investigates the feasibility of domain adaptation of small language models (SLMs) through reasoning distillation from retrieved reasoning traces originating from larger models – a task for which SLMs are the natural target. Second, as future work, we plan to investigate continual pretraining methods for domain adaptation of SLMs, and the size and weight-availability of these models render them appropriate for that line of work. Finally, the relatively approachable hardware prerequisites for running these models contribute to the reproducibility of our evaluations.

The eight models we evaluate are listed below.

- **OLMo-7B** (Allen Institute, 2024) [125]: A 7B parameter model developed as part of the OLMo project to accelerate language model science. It supports a 2K token context window and emphasizes reproducibility.

- **TinyLlama-1.1B-Chat** (TinyLlama Team, 2024) [178]: A compact model trained on 3T tokens with ∼2K context, designed as an efficient baseline for small-scale deployments.

- **Gemma 3 4B-IT** (Google, 2025) [151]: A recent 4B parameter instruction-tuned model with a large 128K context window, representing the newest generation of mid-scale LLMs.

- **SmolLM3-3B** (HuggingFace, 2025) [68]: A lightweight experimental model, evaluated to capture the behavior of smaller 3B-scale instruction-tuned systems.

- **Mistral-7B-Instruct-v0.3** (Mistral AI, 2024) [70]: A highly optimized 7B model with strong efficiency and reasoning performance, featuring a 4K token context size.

- **Llama-3-8B-Instruct** (Meta, 2024) and **Llama-3.1-8B-Instruct** (Meta, 2024) [55]:

Two successive generations of Meta's flagship open models, included to establish strong baselines in the 8B parameter class.

- **Qwen-1.5-14B-Chat** (Alibaba, 2024) [13]: A 14B parameter multilingual model with a 32K context window, representing the upper end of our evaluation range.

Table 4.1 summarizes the parameter counts, release years, and context window sizes of the evaluated models.

Table 4.1: Overview of evaluated SLMs with parameter counts, release years, and context window sizes.

Model Name	Params	Release Year	Context Window
OLMo-7B	7 B	2024	2,048
TinyLlama-1.1B-Chat	1.1 B	2024	2,048
Gemma 3 4B-IT	4 B	2025	128,000
SmolLM3-3B	3 B	2025	32,768
Mistral-7B-Instruct-v0.3	7 B	2024	4,096
Llama-3-8B-Instruct	8 B	2024	8,192
Llama-3.1-8B-Instruct	8 B	2024	32,768
Qwen-1.5-14B-Chat	14 B	2024	32,768

4.4.2 Retrieval Conditions

Each model is evaluated under three retrieval conditions. The first is a no-retrieval baseline in which the model receives only the question and answer options. This condition measures the model's parametric knowledge of the domain. The second is RAG-Chunks, in which the model receives the question augmented with the top-5 most relevant semantic chunks retrieved from the source papers via cosine similarity search over PubMedBERT FP16 embeddings (747 MB total) in a FAISS [41] vector store. This condition measures the

benefit of standard RAG with scientific text. The third is RAG-RT, in which the model receives the question augmented with reasoning traces retrieved from the GPT-4.1 teacher. Each of the three reasoning modes (detailed, focused, efficient) is evaluated independently, yielding three separate RAG-RT conditions: RAG-RT-Detail, RAG-RT-Focused, and RAG-RT-Efficient. This separation allows us to characterize the effect of reasoning verbosity on retrieval-augmented performance.

4.4.3 Evaluation Benchmarks

We evaluate on two benchmarks. The first is the synthetic benchmark of 16,680 MCQs generated by our pipeline (Section 4.2), spanning radiation and cancer biology. This benchmark tests whether models can answer questions derived from the same literature that was used to generate the retrieval corpus. The second is the 2023 ASTRO Radiation and Cancer Biology Study Guide [7], hereafter the Astro exam, an expert-annotated benchmark of 337 questions prepared for the American Society for Radiation Oncology. This benchmark was *not* generated by our pipeline and is therefore independent of both the question generation process and the source corpus. Crucially, this benchmark shares the cancer biology domain with our synthetic benchmark, enabling evaluation of whether reasoning trace retrieval facilitates domain adaptation in small language models. Improved performance would support the claim that reasoning distilled from a teacher via retrieval can enhance a student model within a niche domain.

We exclude two visually dependent questions, yielding 335 evaluable items. We further isolate 189 non-mathematical questions—identified using GPT-5 [120]—to focus on conceptual knowledge, where reasoning traces are expected to provide the greatest advantage. This also avoids confounding retrieval comparisons with the SLMs' limited arithmetic capabilities.

4.5 Results

4.5.1 Synthetic Benchmark Results

We first evaluate models on the synthetic benchmark of 16,680 MCQs in radiation and cancer biology. This large-scale setting allows us to systematically compare baseline performance, retrieval from source document chunks, and retrieval from reasoning traces. The results are presented in Table 4.2 and visualized in Figure 4.4.

Table 4.2: Accuracy of evaluated models on the synthetic benchmark (16,680 MCQs) under baseline, RAG from paper chunks, and the three reasoning-trace (RT) retrieval modes. Best-performing configuration per row is in **bold**.

Model	Baseline	RAG-Chunks	RAG-RT-Detail	RAG-RT-Focused	RAG-RT-Efficient
OLMo-7B	0.380	0.443	0.709	**0.736**	0.720
TinyLlama-1.1B	0.176	0.434	**0.710**	0.699	0.581
Gemma 3 4B-IT	0.745	0.837	0.860	**0.878**	0.873
SmolLM3-3B	0.471	0.803	0.826	0.854	**0.856**
Mistral-7B	0.737	0.839	**0.886**	0.889	0.882
Llama-3-8B	0.830	0.864	0.875	0.892	**0.897**
Llama-3.1-8B	0.819	0.900	0.915	0.902	**0.916**
Qwen-1.5-14B	0.776	0.853	0.913	0.908	**0.914**

Baseline to RAG-Chunks. Chunk retrieval provides a strong first lift across nearly all models. TinyLlama-1.1B-Chat improves from 17.6% to 43.4% accuracy, a relative gain of +147%. SmolLM3 moves from 47.1% to 80.3%, Gemma 3 4B-IT increases from 74.5% to 83.7%, and OLMo-7B improves from 38.0% to 44.3%. These results confirm that chunk retrieval alone provides meaningful domain adaptation for knowledge-intensive scientific MCQA.

RAG-Chunks to RAG-RT. Reasoning-trace retrieval provides a second, consistent jump. TinyLlama reaches 71.0% accuracy with detailed reasoning traces – a roughly 4× improvement over its 17.6% baseline. SmolLM3 reaches 85.6% with efficient traces (+82% relative to baseline). Larger models benefit as well: Llama-3.1-8B-Instruct and Qwen-1.5-14B-Chat both exceed 91% accuracy under their best reasoning-trace configurations.

Reasoning Modes. Across models, all three reasoning modes (detailed, focused, efficient) yield strong improvements with only modest variation among them. The focused and efficient modes often provide the best balance of accuracy and retrieval efficiency, while the detailed mode sometimes trails slightly, likely due to noise from over-elaboration. Notably, Llama-3.1-8B achieves its best performance under the efficient mode (91.6%), suggesting that compact, distilled rationales can be just as effective as full option-by-option analyses – a finding to which we return in Section 4.6.2.

The inverse scaling effect. A striking pattern emerges from Table 4.2: the smaller the model, the larger the relative benefit from reasoning-trace retrieval compared to chunk retrieval. TinyLlama gains approximately 28 percentage points from moving from chunks to detailed traces (43.4% → 71.0%), while Qwen-1.5-14B-Chat gains approximately 6 percentage points moving from chunks to its best RT mode (85.3% → 91.4%). We refer to this as the *inverse scaling effect* of reasoning-trace retrieval: traces are most valuable precisely where they are most needed, i.e., for models with limited parametric capacity that cannot reconstruct the reasoning path from raw evidence alone.

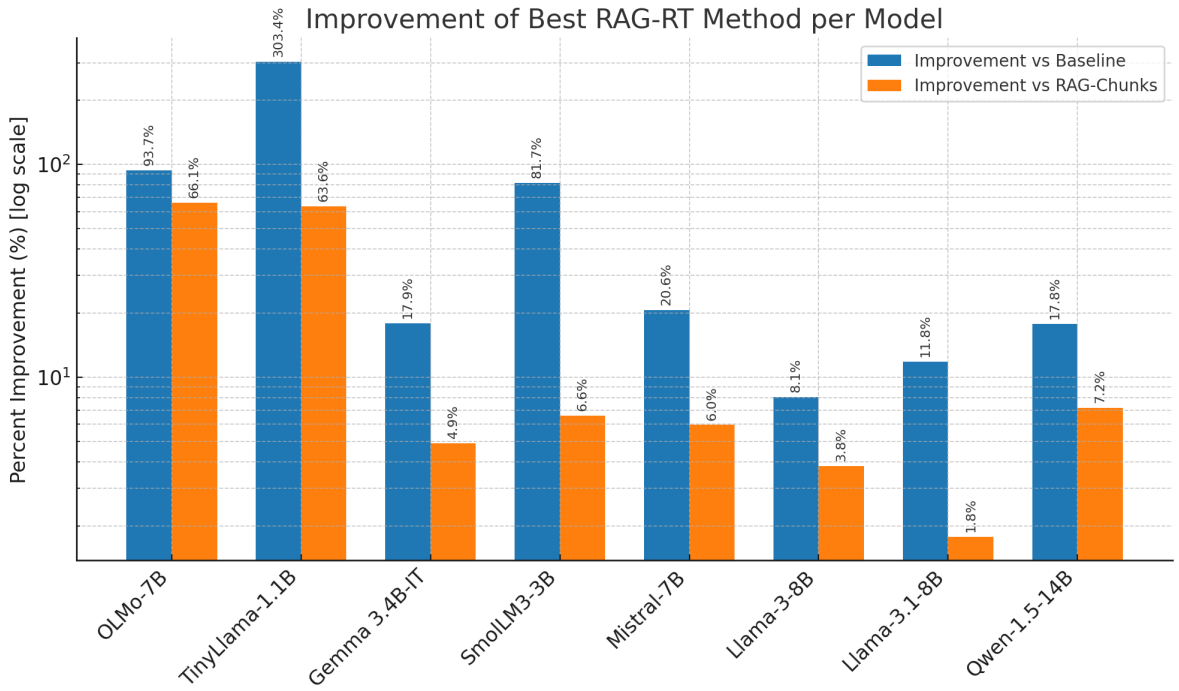

Figure 4.4: Percent accuracy improvement on the synthetic MCQA benchmark, comparing the best reasoning-trace retrieval mode for each model to its baseline performance and to retrieval from source documents.

4.5.2 Astro Exam: All Questions

We next evaluate models on the 2023 ASTRO Radiation and Cancer Biology Study Guide, an expert-written examination that is independent of our pipeline. Across the full set of 335 evaluable questions, reasoning-trace retrieval consistently outperforms baseline and usually surpasses retrieval from source document chunks, though the gains are attenuated relative to the synthetic benchmark. This attenuation is expected: the Astro exam includes questions requiring mathematical computation, diagram interpretation, and clinical judgment that extend beyond the purely knowledge-retrieval setting where reasoning traces are most advantageous.

Gemma 3 4B-IT improves by approximately +25% relative to baseline with reasoning traces, while SmolLM3 and Mistral-7B-Instruct both gain approximately +20% over baseline.

Even the smallest models (TinyLlama-1.1B-Chat, OLMo-7B) see gains of +15–30% over their baseline accuracy. Notably, the relative improvements over RAG-Chunks are smaller and sometimes negative – for example, Llama-3-8B-Instruct's RAG-RT accuracy (0.542) is meaningfully below its RAG-Chunks accuracy (0.674) – suggesting that for some models, direct text retrieval already provides useful context that competes with the reasoning trace. Nevertheless, reasoning traces remain the more stable retrieval source across the model suite. Table 4.3 and Figure 4.5 present the empirical results.

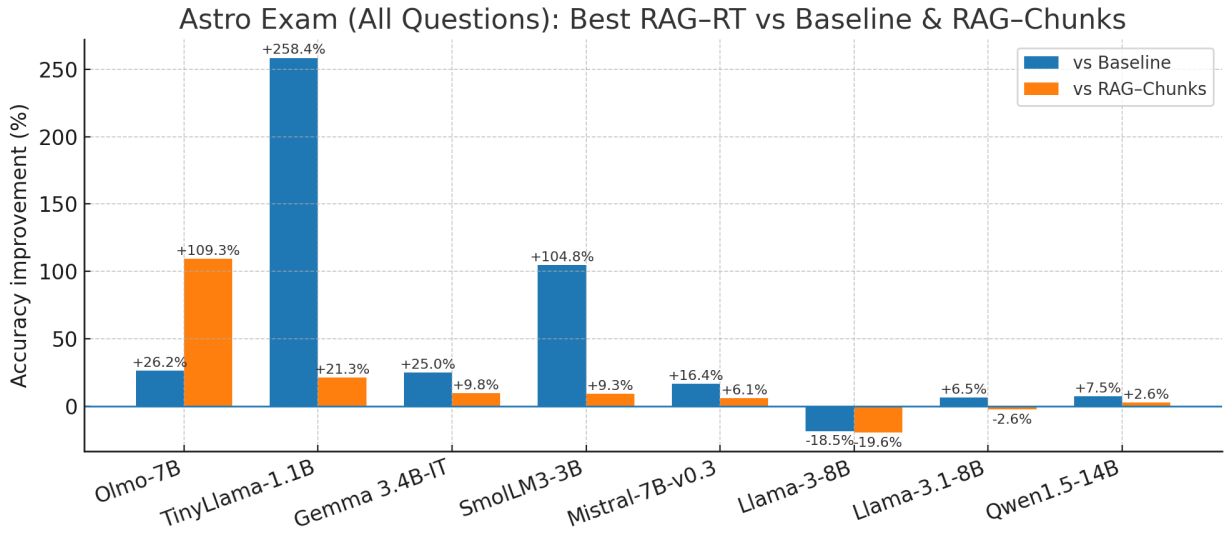

Figure 4.5: Percent accuracy improvement on *all* 335 questions of the 2023 ASTRO Radiation and Cancer Biology Study Guide, comparing reasoning-trace retrieval to both baseline performance and retrieval from source documents for each evaluated model.

Table 4.3: Astro exam (all 335 questions): accuracy for each model under three configurations. Best per row in **bold**.

Model	Baseline	RAG-Chunks	RAG-RTs (best)
OLMo-7B	0.446	0.269	**0.563**
TinyLlama-1.1B-Chat	0.089	0.263	**0.319**
Gemma 3 4B-IT	0.484	0.551	**0.605**
SmolLM3-3B	0.377	0.706	**0.772**
Mistral-7B-Instruct-v0.3	0.494	0.542	**0.575**
Llama-3-8B-Instruct	0.665	**0.674**	0.542
Llama-3.1-8B-Instruct	0.644	**0.704**	0.686
Qwen-1.5-14B-Chat	0.560	0.587	**0.602**

4.5.3 Astro Exam: Non-Mathematical Subset

When the analysis is restricted to the 189 non-mathematical questions, the effect of reasoning traces becomes dramatically more pronounced. All models show positive gains over both baseline and RAG-Chunks. SmolLM3 nearly doubles its accuracy from 46.6% (baseline) to 89.4% (RAG-RT) – a +92% relative gain over baseline and a +19% gain over RAG-Chunks. Gemma 3 4B-IT improves from 54.0% to 80.4%, and Mistral-7B-Instruct improves from 59.8% to 75.7%. The effect is particularly striking for the smallest models: TinyLlama improves from 13.8% to 31.2% – still modest in absolute terms but a 2.3× improvement – and OLMo-7B moves from near-baseline performance to competitive accuracy when supported by reasoning traces. Table 4.4 and Figure 4.6 present the results.

These results demonstrate that reasoning-trace retrieval is most effective for conceptual, knowledge-intensive questions where the trace provides a structured reasoning scaffold that compensates for the model's limited parametric knowledge. For mathematical questions,

where the bottleneck is computational ability rather than knowledge retrieval, the traces provide less benefit – aligning with the intuition that distilled scientific rationales capture high-value domain knowledge but do not substitute for arithmetic capability.

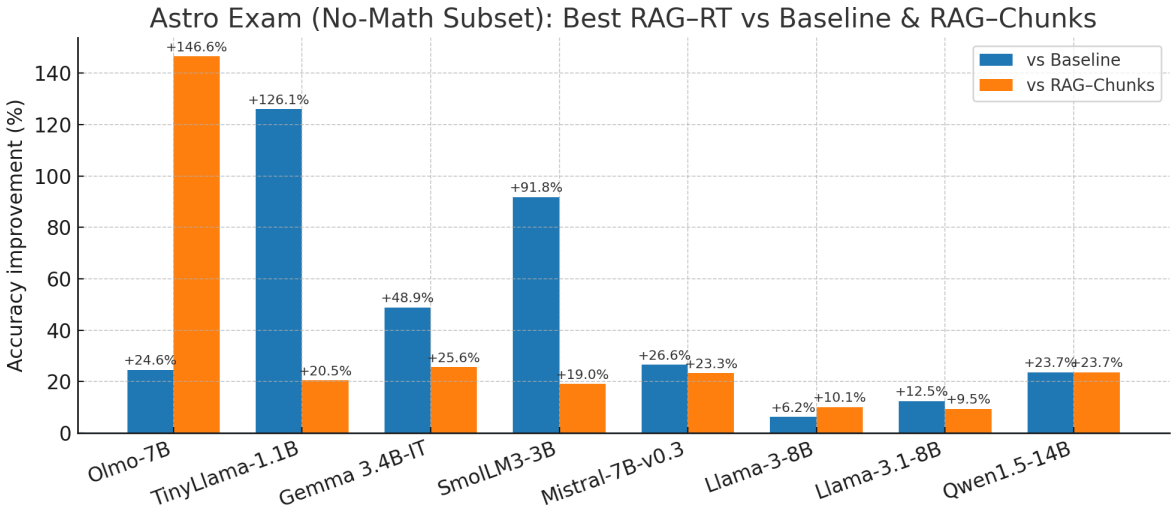

Figure 4.6: Percent accuracy improvement on the *non-mathematical* subset (189 questions) of the 2023 ASTRO Radiation and Cancer Biology Study Guide, comparing reasoning-trace retrieval to both baseline performance and retrieval from source documents for each evaluated model.

Table 4.4: Astro exam (no-math subset, 189 questions): accuracy for each model under three configurations. Best per row in **bold**.

Model	Baseline	RAG-Chunks	RAG-RTs (best)
OLMo-7B	0.471	0.238	**0.587**
TinyLlama-1.1B-Chat	0.138	0.259	**0.312**
Gemma 3 4B-IT	0.540	0.640	**0.804**
SmolLM3-3B	0.466	0.751	**0.894**
Mistral-7B-Instruct-v0.3	0.598	0.614	**0.757**
Llama-3-8B-Instruct	0.757	0.730	**0.804**
Llama-3.1-8B-Instruct	0.762	0.783	**0.857**
Qwen-1.5-14B-Chat	0.667	0.667	**0.825**

4.5.4 Small Models Matching Frontier Performance

Perhaps the most striking result is that several small models, when augmented with reasoning-trace retrieval, match or exceed the performance of GPT-4 on this expert-annotated domain examination. Beattie et al. [14] report GPT-4's overall accuracy on the same 2023 Astro exam (335 questions) at 271/335, or approximately 80.9%. On our evaluations, several SLMs equipped with RAG-RT reach or surpass this figure on the non-mathematical subset: SmolLM3-3B reaches 89.4%, Llama-3.1-8B-Instruct reaches 85.7%, Qwen-1.5-14B-Chat reaches 82.5%, Gemma 3 4B-IT reaches 80.4%, and Llama-3-8B-Instruct also reaches 80.4%. SmolLM3-3B's result is especially noteworthy: a 3B-parameter model, augmented with retrieved reasoning traces, exceeds the reported accuracy of a frontier proprietary model that is roughly two orders of magnitude larger.

This finding has practical implications. Small models are cheaper to run by orders of magnitude, easier to fine-tune for domain adaptation, more suitable for deployment on edge devices or within agentic workflows where latency is critical, and pose fewer data-sovereignty concerns than cloud-hosted API models. The demonstration that these models can achieve frontier-level domain performance through retrieval augmentation – specifically, through retrieval of reasoning traces rather than raw text – suggests a new paradigm for scientific AI deployment.

4.6 Analysis and Ablations

4.6.1 Why Do Reasoning Traces Outperform Raw Chunks?

We hypothesize three mechanisms by which reasoning traces outperform raw chunks as retrieval sources.

Signal-to-noise ratio. Raw scientific text contains substantial amounts of information that is irrelevant to any specific question, e.g., background context, literature reviews, methodological details, caveats, and acknowledgements. A reasoning trace, by contrast, contains only information relevant to answering the specific question at hand. The roughly 15× compression ratio between the trace store (50 MB) and the embedding store of the source chunks (747 MB) is a quantitative reflection of this focused filtering.

Pre-structured reasoning. Raw chunks require the student model to infer the reasoning path that connects the evidence to the answer. Reasoning traces provide this path explicitly, reducing the inferential burden on the student model. For small models with limited capacity for multi-step inference, this pre-structuring is especially valuable.

Implicit cross-document synthesis. When the teacher model generates a reasoning trace, it draws on its full parametric knowledge, which may encompass information from thousands of papers that the student model has never seen. The trace captures this synthesized knowledge in a compact, retrievable form. The student model benefits from this synthesis without needing to perform it independently.

4.6.2 Effect of Trace Type

The reasoning-mode results in Table 4.2 reveal a counterintuitive pattern: the efficient trace (the most compact mode) often performs comparably to or better than the detailed trace

(the most verbose mode). For example, Llama-3.1-8B achieves its best performance under the efficient mode (91.6%), and SmolLM3 likewise peaks under the efficient mode (85.6%). This aligns with the hypothesis that small models benefit from focused, concise scaffolding rather than verbose explanations that may exceed their effective context utilization capacity. Less is more when it comes to reasoning-trace retrieval for small models – a finding that has practical implications for the storage and serving cost of reasoning-trace indices at scale, since efficient traces are an order of magnitude shorter than detailed ones and therefore substantially cheaper to retrieve and consume. Additionally, concise reasoning traces can avoid the pitfall of overfitting the reasoning to a particular questions, thus facilitating better generalization of the distilled knowledge within the adapted niche domain.

4.7 Discussion

The results in this chapter support three conclusions with implications beyond the specific domain of radiation and cancer biology.

Automated benchmark generation is feasible and valuable. The pipeline produces high-quality, provenance-tracked questions that serve as effective evaluation instruments, and the broad performance trends carry over consistently from our synthetic benchmark to the independent expert-annotated Astro exam. The pipeline can be applied to any domain with a sufficient corpus of scientific papers, and its HPC scalability via Parsl enables generation at the scale of hundreds of thousands of questions from millions of papers. Compared to existing efforts in this space, our framework is distinguished by its combination of full automation (PDF parsing through quality-aware filtering), provenance tracking (each question linked to its source chunk and document), and HPC-scale design. SciQ [168] relied on crowdsourcing and is constrained to preselected topics. PubMedQA [74] relies on abstract-based templates for biomedical QA generation. AstroMLab-1 [153] provides 4,425 expert-vetted MCQs in

astronomy but is limited to question retrieval rather than generation. The benchmarks themselves – as we have argued in Section 2.7.2 – are also vulnerable to data contamination as they are static and openly distributed. Our pipeline addresses both gaps by enabling on-demand benchmark generation that is provenance-tracked and re-runnable as new literature appears.

Distillation through retrieval is a viable and advantageous paradigm. The central empirical finding of this chapter – that retrieving reasoning traces enables small models to approach or exceed frontier performance – validates the distillation-through-retrieval mechanism introduced in Section 4.3. Placing this finding in the context of the broader distillation literature surveyed in Section 2.2.3: classical distillation [63] transfers knowledge by modifying student weights to match teacher logits; reasoning-trace distillation [111, 34] transfers knowledge by fine-tuning student weights on teacher reasoning traces; our approach transfers knowledge by making teacher reasoning traces *retrievable at inference time*, leaving student weights untouched. This distinction has concrete advantages for scientific AI:

- **No retraining required.** A new scientific domain can be supported by generating traces for that domain and adding them to the index, without any student fine-tuning. In our experiments, the same student models benefit from radiation biology traces without having been trained on biomedical data.
- **Composable expertise.** Traces from different domains, different teacher models, and different time periods can coexist in the same retrieval index. A student querying the index receives the most relevant trace regardless of its provenance, effectively composing expertise across domains.
- **Updatable knowledge.** As the scientific literature evolves, new traces can be generated and added to the index without discarding or retraining. This stands in direct

contrast to weight-based distillation, where incorporating new knowledge requires retraining and risks catastrophic forgetting of previously distilled capabilities.

- **Auditability.** Every trace is a human-readable document that can be inspected for correctness, bias, and completeness. When a student model produces an answer based on a retrieved trace, the trace itself serves as a verifiable rationale for the answer – a property that is essential for scientific applications where trust and reproducibility are paramount.

The inverse scaling effect that we observe in Section 4.5.1 – smaller models benefiting more from reasoning-trace retrieval – further supports the distillation interpretation. In weight-based distillation, the capacity gap between teacher and student bounds how much knowledge can be transferred: a 1B-parameter student cannot absorb all the knowledge of a trillion-parameter teacher. In retrieval-based distillation, this capacity constraint is relaxed because the knowledge remains external. The student does not need to *store* the teacher's reasoning – it only needs to *follow* it. This explains why even TinyLlama (1.1B parameters) achieves a 4× improvement with reasoning-trace retrieval: the trace provides the reasoning structure that the model's limited parametric capacity cannot generate internally.

Implications for the deployment of scientific AI. The practical implication of these findings is that organizations deploying LLMs for scientific tasks need not choose between expensive frontier models and underperforming small models. Instead, they can deploy small, efficient models augmented with reasoning-trace retrieval, achieving frontier-level domain performance at a fraction of the cost. This has particular relevance for agentic scientific workflows (Section 2.2.2), where thousands of LLM calls per task make frontier-model costs prohibitive. Reasoning-trace retrieval enables small models to serve as domain-specialized agents within these workflows, with the teacher's expertise available on demand through retrieval rather than encoded in expensive parameters.

Ongoing and future work. We are currently scaling the reasoning-trace generation pipeline to Aurora at the Argonne Leadership Computing Facility under the banner of *Project ExaForge* [50], with the target of generating 100 million reasoning traces from 10 million papers spanning the entirety of arXiv and bioRxiv. These traces will be organized by sub-domain with metadata linking each trace to its source, and we plan to explore their use as both retrieval sources and as pretraining and fine-tuning data for LLMs. The central question is whether distillation through retrieval can yield general-purpose scientific reasoning capabilities in small language models when the retrieval index spans diverse scientific domains – a question that would bridge the gap between domain-specific distillation and the emergence of general scientific reasoning.

CHAPTER 5

SWARM RETRIEVAL: REFRAMING RETRIEVAL FROM A STATIC LOOKUP TO AN INTERVIEW PROCESS

This chapter presents Swarm Retrieval, a proposed paradigm for retrieval in which documents are not treated as passive points in a vector space but as document-conditioned agents capable of *qualitatively* evaluating their own relevance to a query, and communicating the associated findings transparently. Unlike the preceding chapters, which report implemented systems and empirical results, this chapter is intentionally forward-looking. It contributes a conceptual framework, systems design, and evaluation agenda motivated by the empirical findings of Chapter 3 and Chapter 4.

The status of the chapter is therefore different from the status of the preceding empirical chapters. Chapter 3 and Chapter 4 demonstrate completed systems, datasets, and experiments. By contrast, the present chapter asks what retrieval would look like if the dissertation's central empirical observation were taken to its logical conclusion: retrieval quality, evidence selection, and reasoning structure often matter more than sheer generator scale. The claims in this chapter should therefore be read as design hypotheses, research questions, and evaluation targets rather than as completed experimental findings.

The central intuition is straightforward: if relevance is not merely geometric proximity, but a query-conditioned judgment about whether a document can contribute useful evidence, then retrieval should not be modeled only as nearest-neighbor search. It can also be modeled as a distributed decision process. In Swarm Retrieval, documents are represented by reusable, KV-cached agents—which we call *paper spirits*—that can accept, reject, abstain, and justify their participation in a generated answer. A judge panel—also consisting of a swarm of LLM agents with a shared KV-cached prefix of adjudication instructions— then adjudicates among the accepting documents and selects the evidence to pass into the generator context.

5.1 Motivation: Beyond Vector Similarity

The preceding chapters demonstrated both the power and the limitations of the prevailing retrieval paradigm. HiPerRAG showed that high-quality dense retrieval can enable general-purpose LLMs to outperform domain-specific models and frontier commercial LLMs on scientific question answering. The MCQA benchmarking work then showed that what is retrieved can matter more than how large the generator is: reasoning traces, when used as retrieval sources, substantially outperform raw text chunks for adapting small models to scientific question answering.

Yet both contributions still operate within the same broad retrieval framework. Documents are encoded as fixed-size vectors, and retrieval is performed by finding vectors near the query in an embedding space. This framework has been extraordinarily useful in facilitating fuzzy search at immense throughput and scale, and Swarm Retrieval is not proposed as an outright rejection of vector search. Rather, it is proposed as a response to three failure modes that become increasingly visible at the scale of scientific corpora: the proxy problem, the scaling problem, and the opacity problem.

5.1.1 The Proxy Problem

Cosine similarity between pooled embeddings is a coarse proxy for the utility of a document to a given query. That said, it is essential to differentiate similarity and utility as distinct qualities. A document that is topically similar to a query, such as a broad review paper on the same subject, may be less useful than a document that is topically narrower but contains the specific empirical result needed to answer the query. Conversely, a document that appears relevant by embedding distance may discuss the same concepts in a context that is irrelevant to the question being asked.

The proxy problem is structural rather than incidental. Pooled embeddings compress rich, multi-faceted documents—spanning methods, results, caveats, definitions, figures, tables,

and arguments—into fixed-size vectors. Pooling operation, whether it is mean pooling, max pooling, or last-token pooling, is inevitably a lossy compression operation that sacrifices sub-document granularity for the sake of practical feasibility. The ColTrast objective in Chapter 3 improves the quality of this compression by aligning token-level and sequence-level representations, but it does not eliminate the information loss inherent in representing a long scientific document as a single point in an embedding space.

5.1.2 The Scaling Problem

As corpus size grows, the density of the embedding space increases. At the scale of millions of articles and tens of millions of chunks, many passages occupy similar neighborhoods. Under these conditions, minute shifts in cosine similarity may determine which passages are retrieved, even when those differences are not semantically meaningful. Retrieval quality can therefore become sensitive to small perturbations in the query, the chunk boundary, the encoder, or the index configuration. This sensitivity is not merely an artifact of noise but reflects a fundamental limitation of single-vector embeddings, where the number of distinct top-k retrieval sets that can be represented is inherently bounded by the embedding dimension, making certain relevance distinctions impossible even under ideal optimization [169].

This scaling problem does not entirely debunk the merit of dense retrieval. In fact, the results of Chapter 3 show the opposite: that dense retrieval can be highly effective when the parsing, chunking, encoding, and indexing pipeline is carefully engineered as an end-to-end unified system. Our argument here is more precise: as scientific corpora grow, the space of near-matches becomes crowded, and the distinction between a useful evidence-bearing passage and a merely topical distractor becomes harder to express through distance alone.

5.1.3 The Opacity Problem

Vector retrieval also produces limited—if any—explanation of its selections, and it fails opaquely. When a RAG system retrieves five chunks and presents them to a generator LLM, the generator receives the retrieved text but not an account of why those chunks were selected. Conversely, systems that implement dense retrieval on vector similarity are incapable of explaining the reason for rejecting a candidate for retrieval, even in high-stake cases where this property is essential. The retrieval decision is determined by geometric proximity in a high-dimensional space. Navigating that geometry is useful operationally and scalable computationally, but it is not directly interpretable by the user or by the downstream generator.

This opacity is especially problematic for scientific applications. A scientist does not merely need text that is topically nearby. They need to know whether the retrieved evidence is relevant, whether it supports or contradicts the claim at hand, and whether it is specific enough to justify downstream reasoning. In this setting, the absence of an explicit relevance rationale is not a cosmetic limitation. It affects auditability, trust, and the ability to diagnose failures.

5.1.4 The Insight from Reasoning Traces

The reasoning-trace experiments in Chapter 4 provide the key motivation for Swarm Retrieval. When scientific questions are paired with reasoning traces produced by a stronger model, small generators can use those traces as in-context scaffolding. Under most evaluation scenarios, we report that retrieved reasoning traces are more useful than retrieved raw chunks (chapter 4). This suggests that the value of retrieval lies not only in the retrieved artifact itself, but in the reasoning structure that connects that artifact to the question. In other words, ultimate performance of the generator does benefit from an explanation as to why a given unit of reference information qualified for an allocation in its limited context

window given the specific prompt it has to answer.

Swarm Retrieval extends this observation from answer generation to evidence selection. If reasoning traces help a generator use evidence, then retrieval itself should include a reasoning step: documents should not only be retrieved because they get encoded similarly to a query, but because they can defend their case for why they are relevant to it. The rest of this chapter develops that idea as a forward-looking research program.

5.2 The Swarm Retrieval Paradigm

Swarm Retrieval reframes retrieval as an adjudicated interview rather than a silent lookup. In the interview metaphor, documents are not passive records to be scanned. They are participants that can make structured claims about their own relevance. A separate judge panel then evaluates these claims and selects the documents whose relevance arguments are most useful for the downstream task.

The metaphor is useful only insofar as it clarifies the technical primitive. The proposed retrieval unit is not merely an embedding vector. It is a document-conditioned decision-maker: a reusable model state bound to a document, a relevance rubric, and an output schema. We refer to this unit as a *paper spirit*. The term is intentionally metaphorical, but the underlying object is concrete: a KV-cached document prefix paired with instructions for producing structured relevance judgments.

The paradigm consists of three components:

1. **Paper Spirits:** Each document, section, or chunk in the corpus is associated with a small language model agent whose cached prefix contains the document content and self-evaluation instructions shared by the entire swarm. The paper spirit is responsible for emitting a structured relevance decision with evidence pointers and a short justification. The schema for this self-assessment is depicted in Figure 5.3

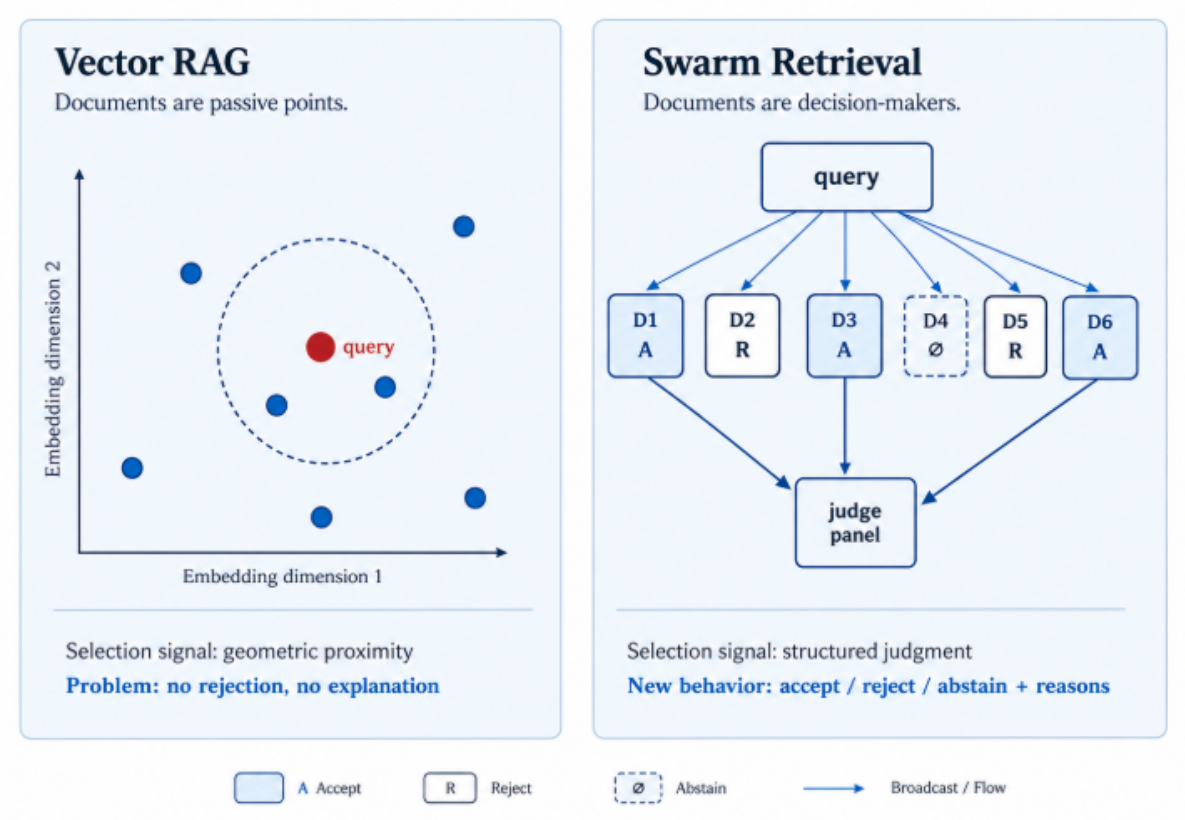

Figure 5.1: Agency shift from Vector RAG to Swarm Retrieval. In Vector RAG, documents are passive points selected by geometric proximity. In Swarm Retrieval, documents act as query-conditioned decision-makers that can accept, reject, abstain, and justify their relevance.

2. **Query Broadcast:** At query time, the user query is broadcast to a swarm of paper spirits. In a pure-swarm version, the query is broadcast to all spirits in a small or medium-sized corpus. In a hybrid version for large-scale production-level deployment, a conventional retriever first narrows the candidate pool based on existing faster search methods like sparse or dense retrieval, and the query is then broadcast only to the selected candidate spirits.

3. **Judge Panel:** A separate set of judge models evaluates the self-assessments produced by accepting spirits. The judges rank the candidate documents based on specificity, faithfulness, evidence quality, and task relevance, producing the final top-k context. A judge can be powered by either the same small model that drives a paper spirit, or a different one. In the former setup, what differentiates a judge from a paper spirit is simply the prefixed system prompt (instruction) they have cached.

The output of Swarm Retrieval is therefore not merely a ranked list of document IDs—though such a ranking can be produced either by demanding a Likert-scale ranking from judges or by validating vector similarity on the selected documents. Rather, it is a ranked list of documents accompanied by explicit relevance decisions, confidence estimates, evidence spans, and justifications. These artifacts can be passed to the generator model alongside the retrieved text, giving the generator a more interpretable account of why the evidence was selected. Moreover, these substantiating artifacts can be elevated to the user for human-in-the-loop supervision, passing the accountability of final call to a human at high-stakes decision-making scenarios such as clinical applications.

5.3 KV-Cached Document Agents

The primary objection to Swarm Retrieval naturally challenges its computational feasibility. Running a language model, albeit a small one, over every document for every query would be prohibitively expensive under a naive implementation. KV caching points towards a possible systems path around this objection by amortizing document processing across queries. In this design, the cache is not merely an inference optimization; it becomes part of the retrieval substrate itself.

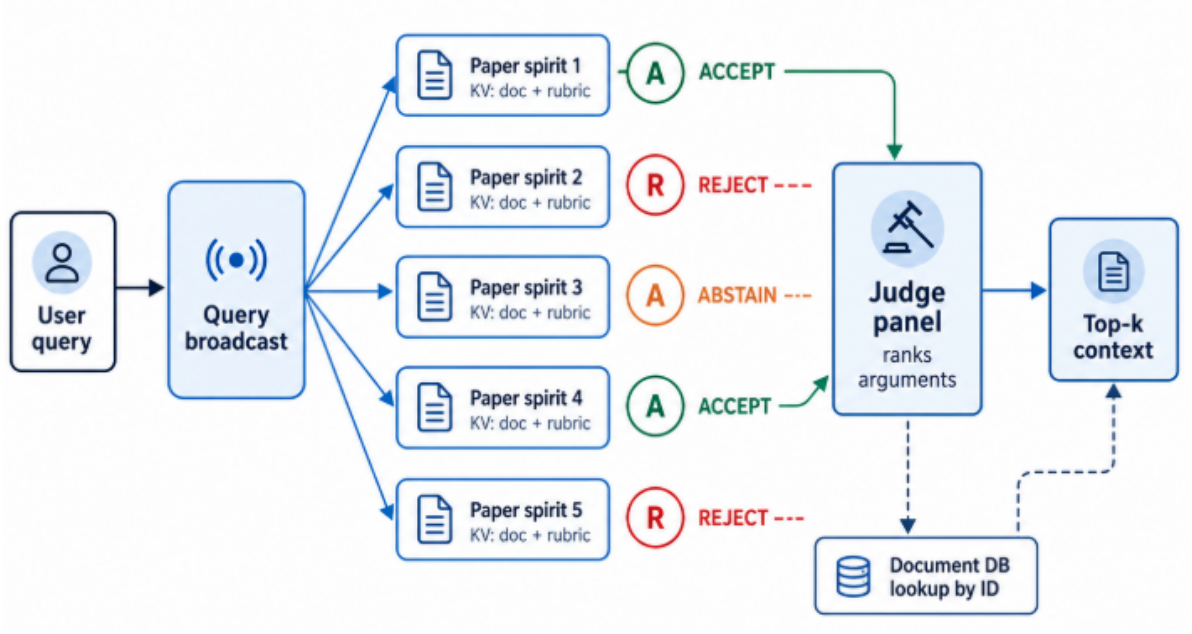

Figure 5.2: Overview of the Swarm Retrieval architecture. A query is broadcast to KV-cached document agents, or paper spirits, which emit structured accept, reject, or abstain decisions. Accepted documents forward reasoned self-assessments to a judge panel, which selects the top-ranked evidence for downstream generation.

5.3.1 KV Cache Mechanics

In a standard Transformer decoder, the self-attention mechanism computes key and value projections for each token in the input sequence at each layer. For a sequence of length n and a model with L layers, each with key and value dimension d_k, the KV cache stores the intermediate key and value states needed for subsequent decoding. Once these states are computed, the model can generate new tokens by attending to the cached prefix without recomputing the full document prefix from scratch.

In Swarm Retrieval, the cached prefix for each paper spirit consists of two components:

- the document content, consuming n_{doc} tokens; and
- the self-evaluation instructions, consuming n_{instr} tokens.

At query time, the query tokens n_q are appended to the cached prefix, and the model generates a structured self-assessment of length n_{gen}. The intended advantage is that the

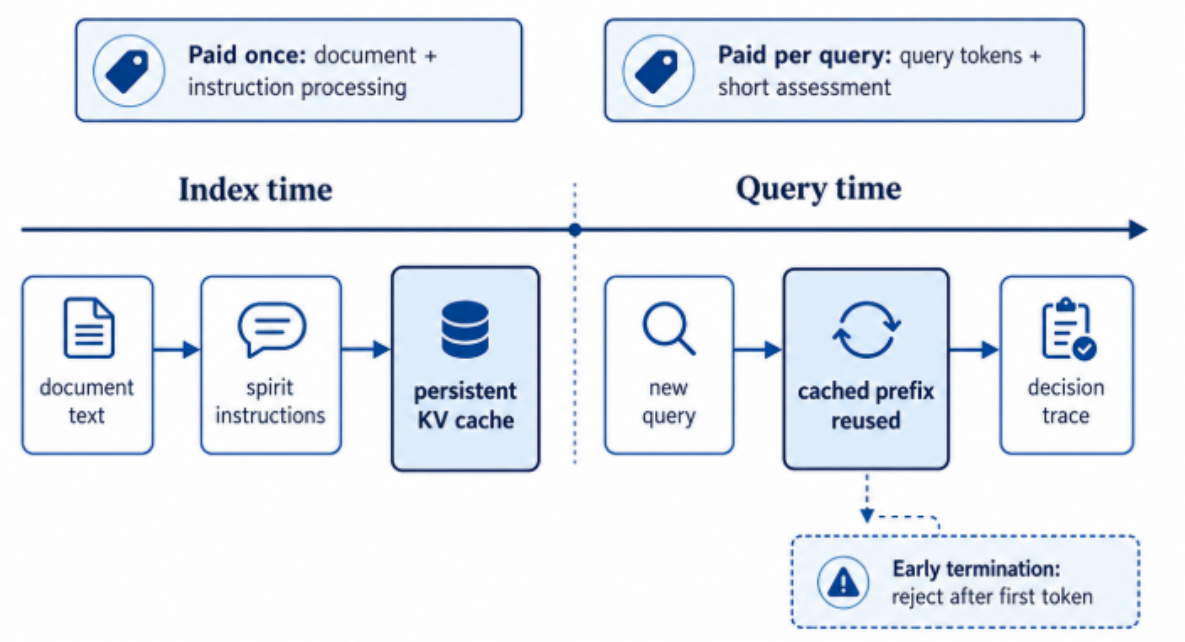

Figure 5.3: KV cache as the retrieval substrate. Document content and self-evaluation instructions are processed once at index time and stored as a persistent KV cache. At query time, the system pays only for the appended query and short decision trace, with early termination for rejected documents.

document processing cost, $n_{\text{doc}} + n_{\text{instr}}$, is paid once during indexing and reused across subsequent queries. The query-time cost is then dominated by the appended query and the short decision trace rather than by repeated full-context processing. Collectively, KV-cached prefixes of all paper spirits in a corpus instantiate a Swarm Retrieval index for that corpus.

This design equips a Swarm Retrieval index with favorable features for practical deployment. First, Swarm Retrieval indexes are transferable across compute fabrics provided the deployment targets of interest can handle the same floating point precision used in the KV-cache. Second, a Swarm Retrieval index benefits from the adaptability that is characteristic of RAG systems, i.e., individual paper spirits can be added or removed post-hoc throughout the lifecycle of the index. Third, a dedicated compute infrastructure can be swiftly repurposed to host a different corpus by simply swapping these KV-caches, an operation that is highly optimized in today's LLM deployment research landscape [96, 83, 152]. Finally, Swarm Retrieval indexes can be composed to form compound indexes.

5.3.2 Early Termination

A second proposed efficiency mechanism is early termination. In large corpora, most documents are irrelevant to any given query. If a paper spirit's first generated decision token indicates rejection or abstention—by design, it will—generation can be preempted immediately, avoiding the cost of decoding a full justification token-by-token. This is a design hypothesis, not a measured result in this dissertation. Crucially, the system can be configured to avoid this preemption to cater for applications that require, and can afford, a justification for every rejection.

As an illustrative calculation, if 95% of documents are irrelevant to a typical query and can be rejected after a single decoding step, early termination would reduce the amount of full self-assessment generation by approximately 20× relative to a system that generates full justifications for every document. This estimate is a systems hypothesis which we plan to be validate by measuring latency, throughput, memory use, and retrieval quality under different rejection thresholds.

5.3.3 Why Small Models May Be Sufficient (and Necessary)

A central design hypothesis of Swarm Retrieval is that the self-evaluation task assigned to paper spirits is narrow enough to be handled by small language models. Structured relevance judgment conditioned on a document and a query imposes substantially lower computational and representational demands than open-ended scientific discovery, and research on small language models as potential workhorses of agentic AI is gaining traction [15]. Our empirical results of Chapter 4 provide indirect support for this hypothesis: small models can perform surprisingly well on structured scientific question-answering when supplied with retrieved context. Whether this transfers to document-level relevance self-assessment remains an open empirical question we aim to investigate.

Small models (SLMs) are attractive in this setting for three reasons. First, they reduce the memory footprint of maintaining many persistent agents. The task calls for a manageable upper-bound on the context length to hold a single document, concise instructions, and a structured output designed for brevity. Second, they make high-throughput decision generation more plausible simply because great numbers of these SLM instances can be run in parallel thanks to their low memory footprint, increasing throughput as a result. Third, they are easier to specialize for narrow output schemas such as `ACCEPT`, `REJECT`, `ABSTAIN`, confidence, reason code, and evidence span. Moreover, size of SLMs position them well for fine-tuning routines aiming to specialize them for the general Swarm Retrieval task we have been defining in this chapter. These advantages are contingent on a thorough calibration: an SLM-based paper spirit that always accepts, confidently rejects relevant documents, or fabricates evidence would be useless or even catastrophic. Viability of the approach hinges on whether SLMs can be made faithful, calibrated, and non-degenerate under realistic retrieval workloads.

5.4 Self-Evaluation and Argumentative Retrieval

In Swarm Retrieval, relevance is not computed exclusively by a distance metric, but is expressed as a structured decision narrative. Each paper spirit receives a query and emits an output object containing a relevance decision, a confidence estimate, evidence pointers, and a short explanation. The retrieval system then selects among these arguments rather than selecting among silent vectors.

5.4.1 Paper Spirit Self-Assessment

A paper spirit's self-assessment should be strict enough to support automatic evaluation and constrained enough to prevent latency-inducing free-form overgeneration. A candidate schema is depicted in Figure 5.4

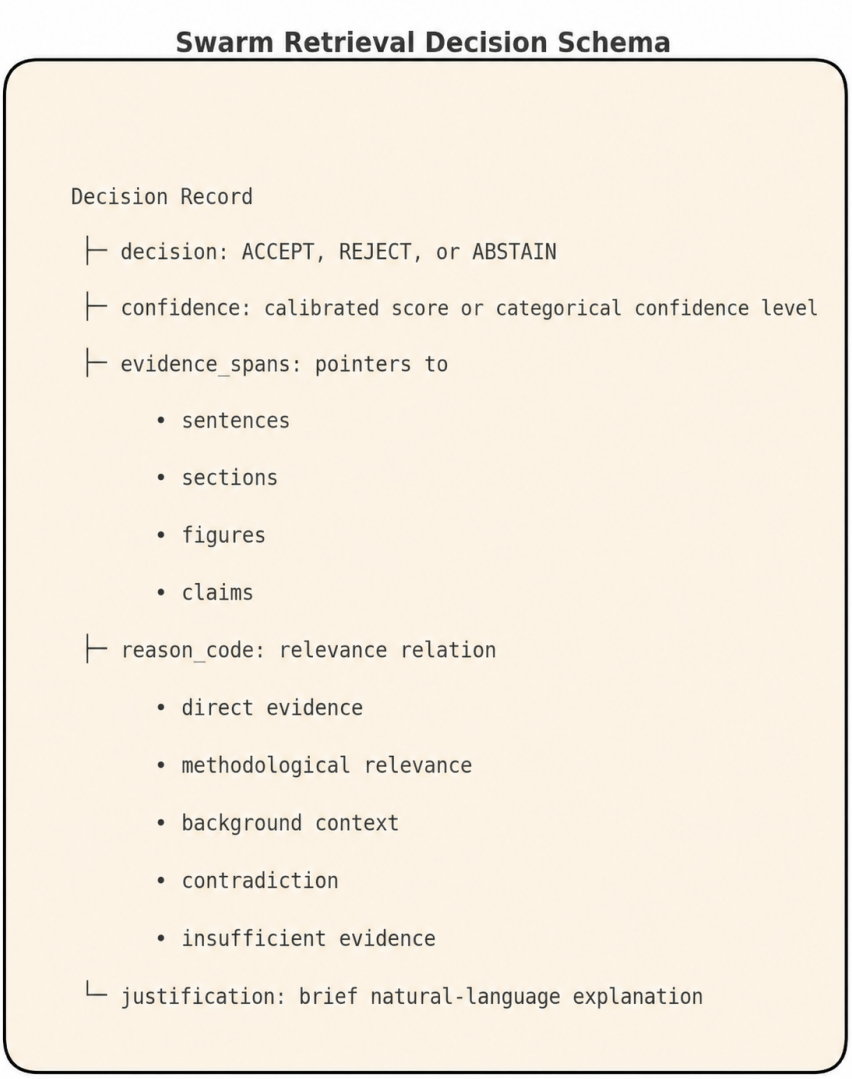

Figure 5.4: Schema proposed in this figure aims to standardize the relevance judgement of each paper spirit. The schema is designed with the purpose of promoting robustness, reproducibility, acceptable latency, and favorable overall computational efficiency.

The inclusion of `ABSTAIN` is important and is a unique advantage of Swarm Retrieval over its predecessors. Standard retrievers are forced to return a ranking even when the corpus does not contain useful evidence. Swarm Retrieval makes uncertainty—for selection, rejection, and abstention alike—explicit. More importantly, it represents a retrieval system that can choose to return *no response as a response* in and of itself, and clearly express its reasoning behind such decision. Indeed, there exist use cases in which this quality is not

merely a tangential addition but a non-negotiable requirement: in clinical decision support, for instance, a system that confabulates a source for a drug interaction or dosage guideline is strictly worse than one that abstains, since a clinician can seek further consultation but cannot un-act on misinformation.

5.4.2 Judge Adjudication

It is possible to envision a system where paper spirits are aligned with human-experts' judgment on relevance through instruction fine-tuning and reinforcement learning [165, 122]. Given such a system, it is conceivable to argue for the redundance of the judge panel since the paper spirits can be instructed to output quantitative scores, e.g. out of 100, that align with experts. We acknowledge that as a potential future direction, and a notable alternative.

As we propose it in this chapter, the judge panel is introduced as a security layer to prevent self-promotion and to separate document-level advocacy from final selection. If paper spirits are allowed to argue for themselves, then the system must also guard against degenerate behavior in which many documents claim relevance. The judge panel receives the self-assessments from accepting spirits and ranks them based on the specificity and credibility of the relevance argument.

In the simplest design, the judge receives the paper spirit's structured output and document metadata. A stricter design may also provide the cited evidence spans so that the judge can verify whether the justification is grounded in the source text. Several adjudication strategies are possible: a single judge, majority vote among multiple judges, confidence-weighted aggregation, or a calibrated scoring model trained on relevance labels. Determining which of these mechanisms is necessary is part of the future evaluation agenda.

Table 5.1: Comparison of retrieval paradigms.

Feature	**Vector RAG**	**Swarm Retrieval (proposed)**
Selection signal	Distance or similarity	Structured judgment
Document role	Passive point	Document-conditioned agent
Adaptability	Re-embedding or reranking	Rubric and judge modification
Output	Ranked text chunks	Chunks plus decisions and justifications
Uncertainty behavior	Forced ranking	Reject or abstain as first-class outputs
Failure mode	Topical but unhelpful retrieval	Over-acceptance or miscalibrated rejection
Index type	Embedding vectors	Embeddings and/or persistent KV caches
Interpretability	Limited	Designed-in relevance rationales

5.5 Comparison with Existing Paradigms

Swarm Retrieval differs from existing retrieval paradigms in the locus of agency and the nature of the relevance signal. In standard dense retrieval, the document is represented as an embedding and selected by proximity. In LLM reranking, an external model reviews a candidate set and reranks the documents. In agentic RAG, an agent may rewrite queries, plan retrieval steps, and orchestrate tools, but the documents themselves remain inert and passive. Swarm Retrieval places part of the decision process inside the corpus: documents become query-conditioned participants in evidence selection.

This distinction does not imply that we propose Swarm Retrieval as a replacement for the existing retrieval methods. Rather, we envision it as a complementary addition to a hybrid system that features the state-of-the-art in sparse, dense, and Swarm Retrieval. Each of these approaches has irreplaceable strengths: sparse retrieval methods such as BM25 [133] and TF-IDF [149] are deterministic and computationally efficient, but their reliance on exact lexical overlap makes them brittle to synonymy, paraphrase, and morphological variation, limiting their applicability in domains where surface-form consistency between queries and documents cannot be assumed. Dense retrieval methods address this shortcoming by the innately fuzzy nature of distance-based semantic search which is opaque in both its successes and failures. Swarm Retrieval complements sparse and dense retrieval with its transparent

outputs and versatile design, yet even with the most ingenious engineering innovations, it is prone to be orders of magnitude slower than both.

We introduce a hybrid retrieval architecture that leverages the advantages of each underlying retrieval method while simultaneously mitigating their respective weaknesses. Accordingly, vector retrieval, BM25 [133], or ColBERT [80, 136] serve as the initial high-throughput filter to identify a candidate pool from the entire corpus (of potentially billions of text chunks). Swarm Retrieval can then act as a second-stage decision system that emphasizes rejection, abstention, polarity control, and reasoned justification. While the hybrid approach appears to be the most practical option for real-world deployment, the standalone configuration remains experimentally significant—not because it is expected to be the primary deployment strategy at the scale of millions of documents, but because it evaluates whether the proposed primitive can perform retrieval without depending solely on nearest-neighbor gating.

5.6 Evaluation Framework

Because Swarm Retrieval is proposed here as a future research direction, its evaluation must be designed to test whether the paradigm provides behavior that is not reducible to ordinary reranking. The goal is not merely to improve a standard ranking score, rather, it is to determine whether document-conditioned judgment provides useful retrieval behaviors that are difficult to obtain from similarity alone: explicit rejection, calibrated abstention, evidence-grounded justification, robustness to distractors, and improved handling of ambiguity. Towards this direction, we propose to evaluate our approach in four axes: retrieval quality, reasoning quality, robustness, and systems-level efficiency (Figure 5.5).

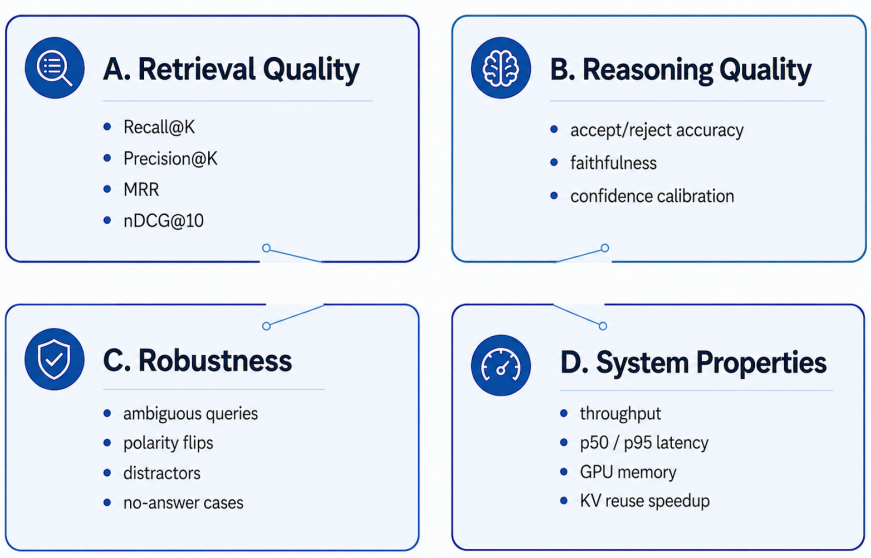

Figure 5.5: Conceptual evaluation framework organized along four complementary axes: retrieval quality (ranking performance), reasoning quality (faithfulness, selectivity, and evidence grounding), robustness (behavior under ambiguity, distractors, and polarity shifts), and system-level efficiency (throughput, latency, and KV-cache reuse).

5.6.1 Axis A: Retrieval Quality

Standard information retrieval metrics remain applicable in evaluating Swarm Retrieval. Recall@K, Precision@K, MRR, and nDCG@10 permit direct comparison of Swarm Retrieval against BM25, dense retrieval, late-interaction retrieval, hybrid retrieval, and LLM reranking baselines, and address the threshold question of whether the proposed system retrieves relevant documents at competitive quality. These metrics are nonetheless insufficient on their own. A marginal improvement in nDCG at substantially greater computational cost would not justify the added system complexity, regardless of conceptual novelty. The stronger empirical case for the paradigm must therefore come from diagnostic evaluation settings in which the capacity for explicit, interpretable decision-making is itself the quantity of interest.

5.6.2 Axis B: Reasoning Quality

A unique evaluation axis for Swarm Retrieval is the quality of the paper spirits' self-assessments. At minimum, this includes Faithfulness: whether the justification is grounded in the source document; Acceptance accuracy: whether accepted documents are truly useful for the query; Rejection accuracy: whether rejected documents are truly irrelevant; and Evidence quality: whether cited spans actually support the stated relevance decision.

A foreseeable catastrophic risk is over-acceptance, i.e., if paper spirits frequently claim relevance without evidence, then the system degenerates into a costly reranker with redundant textual output of no value. We envision true selectivity in this context to lead to a retrieval system that can indicate not only the relevant documents, but also present a interpretable explanation to accompany and defend that judgement. Ultimately, the system holds accountability for each document that selected for or excluded from the context.

5.6.3 Axis C: Robustness

The most important evaluation settings are those where similarity retrieval is known to be fragile. These include ambiguous queries, topical distractors, polarity flips, and no-answer cases. For example, a topically similar document may refute rather than support a claim. A nearest-neighbor retriever may return it as relevant, while a document-conditioned agent should ideally identify the polarity mismatch.

The hypothesis is that Swarm Retrieval should degrade more gracefully under these conditions, producing calibrated rejection or abstention rather than confident false positives. This hypothesis should be tested directly against dense retrieval, late-interaction retrieval, and LLM reranking baselines. The relevant diagnostic metrics include false accept rate, false reject rate, abstention precision and recall, intent coverage, and polarity error rate.

5.6.4 Axis D: System Properties

The final evaluation axis is systems feasibility. Even if Swarm Retrieval provides better diagnostic behavior, it is only useful if its cost can be bounded and justified within the context of its specific deployment scenarios. The key systems measurements are:

- **Throughput:** documents evaluated per second and queries answered per second.
- **Latency:** p50 and p95 latency, with stage-level breakdowns for broadcast, spirit assessment, adjudication, and context assembly.
- **Memory footprint:** peak memory per cached spirit and total memory as the swarm size grows.
- **KV reuse speedup:** measured speedup of cached-prefix inference relative to uncached document-conditioned inference.
- **Amortized indexing cost:** the cost of building persistent document states relative to expected query volume.

These measurements determine where the paradigm is practical. A pure-swarm design may be feasible for small curated corpora, while large-scale scientific corpora may require hybrid architectures we discussed in section 5.5 along with batching, and budgeted evaluation.

5.7 Discussion

Swarm Retrieval is best understood as a research agenda rather than a completed retrieval system. In the preceding chapters we showed that retrieval quality and reasoning structure can substantially affect downstream scientific question answering. In this chapter, we ask

what retrieval would look like if that observation were taken to its logical conclusion: if relevance is not merely geometric proximity, but a query-conditioned judgment about whether a document can contribute useful evidence.

Several open questions define the next stage of the work.

Does reasoning-driven selection outperform similarity-based selection? Our central hypothesis is that structured relevance reasoning captures aspects of document utility that embedding distance and lexical overlap cannot. Merit of Swarm Retrieval hinges on rigorously evaluating this hypothesis. We do not discard the possibility that the additional compute spent on per-document reasoning as a prerequisite for Swarm Retrieval would be better spent on stronger encoders, better chunking, or more sophisticated reranking.

What is the right granularity for paper spirits? A spirit could represent a full paper, a section, a paragraph, a semantic chunk, or even a claim-level unit. The trade-off is that finer granularity improves precision at the computational cost of query broadcast, judge panel adjudication, index building, maintenance, and storage. Conversely, coarser granularity reduces system overhead but may force a spirit to reason over too much content.

Where should final authority reside? If documents argue for themselves, the system requires a mechanism for adjudication. Here we are faced with two options: to trust paper spirits directly or to introduce independent judges. We hinted at a third option in subsection 5.4.2: to combine self-assessment with retrieval scores, metadata, citation signals, or external verification. The correct design depends on the failure modes observed in controlled experiments which are planned.

Can the system scale to scientific-corpus size? KV caching and early termination are proposed as mechanisms for making the system feasible, but they do not remove the

engineering challenge associated with handling KV-caches for every key-value pair in a full-size scientific corpus of millions of papers. We conceive several approaches to the problem. First, we can perform preprocessing on the corpus to summarize its content before KV-caching, eliminating noise. We have built the infrastructure to perform this at the scale of hundreds of millions of documents [50] on the Aurora exascale system. Second, we will explore efficient attention mechanisms [32, 31, 126] and custom scientific tokenizers [17, 30, 56] to reduce the necessary KV-cache. Beyond KV-cache management, scheduling large numbers of short generations for paper spirits and judges, and calibrating rejection thresholds at scale remain open systems questions.

Is Swarm Retrieval a new primitive or a reranking variant? This distinction warrants experimental rather than rhetorical resolution. A pure-swarm configuration evaluated on a representative subset would establish whether document-conditioned agents are capable of performing retrieval in the absence of an ANN candidate gate. Hybrid configurations would subsequently determine whether the primitive affords diagnostic behaviors that exceed those of conventional LLM reranking—in particular, principled abstention, polarity control, and intent coverage.

5.8 Conclusion

In this chapter we proposed Swarm Retrieval as a forward-looking framework for reasoning-driven document selection. Our proposal follows from the trajectory of the dissertation. In Chapter 3 we showed that high-performance retrieval infrastructure can substantially improve scientific question answering without changing the generator model. In Chapter 4 we showed that reasoning traces can serve as powerful retrieval artifacts for adapting small models. Swarm Retrieval extends these findings into a broader question: should retrieval itself become a reasoning process?

The immediate research question is not whether Swarm Retrieval should replace vector retrieval outright. The question is whether it can expose behaviors that vector retrieval does not naturally provide: explicit rejection, calibrated abstention, evidence-grounded justification, and robustness to topical but misleading distractors. If these behaviors can be demonstrated at acceptable cost, then the future of scientific retrieval may not lie only in finding the closest point in an embedding space, but also in asking the corpus to explain why it should be trusted.

CHAPTER 6

CONCLUSION

This dissertation set out to investigate whether the design of retrieval-augmented reasoning systems, when informed by the high-performance computing infrastructure on which they run, could meaningfully change what scientific AI is capable of. The central claim we aimed to advance boils down to:

> *The path to reliable scientific AI lies not in larger models but in co-designing retrieval, reasoning, and HPC infrastructure as a unified system.*

The three preceding contributions—HiPerRAG (Chapter 3), the automated MCQA benchmark generation pipeline with reasoning-trace retrieval (Chapter 4), and Swarm Retrieval (Chapter 5)—were each formulated to test a different facet of this claim. Together, they support a unified position about how scientific AI should be built and where its near-term progress is most likely to come from.

6.1 Summary of Contributions

HiPerRAG. Chapter 3 presented HiPerRAG, a distributed RAG workflow that indexes 3.6 million scientific articles across three leadership-class supercomputers. The chapter demonstrated that the co-design of retrieval algorithms with HPC infrastructure can yield scientific RAG systems that outperform both domain-specific models and frontier commercial LLMs on scientific question answering, and do so without modifying the generator's weights. The discussion featured dedicated evaluation results for the key components of HiPerRAG. Specifically, the Oreo parser delivered a 4.5× throughput improvement and a 94.6× improvement in FLOP utilization over Nougat, while supporting 20 categories of document elements rather than three. Similarly, the ColTrast objective provided a principled way of

jointly finetuning encoders at the token and sequence level for scientific retrieval. The most consequential empirical finding of the chapter was that a Mixtral-8x7B model retrieving with PubMedBERT from our Protein Literature Corpus surpassed the retrieval-free performance of GPT-4 (a frontier model at the time) on the PubmedQA benchmark. This finding inverted the conventional wisdom that improving LLM performance on scientific tasks requires either larger models or domain-specific pretraining, and it established retrieval quality as one of the salient contributors to downstream accuracy in knowledge-intensive scientific tasks.

Automated MCQA generation and reasoning-trace retrieval. Chapter 4 extended this thesis from *retrieval over text* to *retrieval over reasoning*. The chapter introduced an automated MCQA benchmark generation pipeline that produces provenance-tracked questions at HPC scale, addressing the structural problem of benchmark contamination by enabling topically relevant evaluation sets on demand. Applied to radiation and cancer biology, the pipeline yielded 16,680 questions from 22,548 open-access articles. Beyond the pipeline itself, the chapter introduced *distillation through retrieval*. Rather than fine-tuning small models on teacher reasoning traces, we stored those traces in a vector index and retrieved them at inference time. The overarching empirical result is that TinyLlama-1.1B improves from 17.6% to 71.0% accuracy on the synthetic benchmark, and 8B-parameter models exceed 91% accuracy under their best reasoning-trace configurations. The *inverse scaling effect* observed in this setting is, to our knowledge, a previously undocumented property of retrieval-augmented reasoning. We therefore posit that traces are most valuable precisely where they are most needed, i.e., for models with limited parametric capacity that cannot reconstruct the reasoning path from raw evidence alone.

Swarm Retrieval. Chapter 5 took the trajectory of the preceding chapters to its logical conclusion. If retrieval quality matters more than generator scale, and if reasoning structure matters more than raw text, then retrieval should evolve beyond a static lookup and be

reimagined as an active dialogue between the documents and the query. Swarm Retrieval reframes documents not as passive points in a vector space but as document-conditioned agents that can accept, reject, abstain, and justify their participation in a generated answer. Our discussion in this chapter is intentionally forward-looking as it contributes a conceptual framework, a systems design built on KV-cached document agents and early termination, and an evaluation agenda organized around four axes (retrieval quality, reasoning quality, robustness, and systems feasibility). It is worth noting that we do not aim to replace vector search with Swarm Retrieval, which would admittedly be infeasible given the computational cost and latency in favor of vector search. Rather, we propose a complement to the existing paradigm which equips retrieval systems with the interpretability and versatility required in high-stakes domains like medical and nuclear sciences.

6.2 Limitations

We register several limitations with which the empirical claims of this dissertation should be read. First, although HiPerRAG indexes 3.6 million articles, this represents approximately 1.8% of the estimated 200 million scientific papers in existence, i.e., the system has not yet been validated at the scale of the full scientific corpus. Second, retrieval throughout this work is text-only, which admittedly excludes crucial modalities of dissemination such as scientific figures, tables, and equations. Third, the reasoning-trace experiments rely on a single frontier teacher model (GPT-4.1). Consequently, whether the observed gains generalize across teacher models, and whether trace quality is sensitive to teacher choice, remains to be characterized in subsequent work which we intend to do. Fourth, the ColTrast objective carries the risk of overfitting on the token distribution that comprises its training data. This trade-off between custom-tailored, query-aware encoders and their more generic and thus generalizable counterparts remains an active research problem. Fifth and finally, Swarm Retrieval is presented as a research agenda rather than a completed system, and its viability

is contingent on empirical results that remain to be obtained.

6.3 Broader Implications

The findings of this dissertation carry implications beyond the specific systems it presents.

The frontier of scientific AI is no longer defined exclusively by model scale. A small model with access to high-quality retrieval and reasoning traces can match or exceed a frontier model operating without retrieval, and do so at a small fraction of the inference cost, with full auditability, and without retraining for each new domain. As scientific enterprise globally evolves toward an agentic AI-assisted future, workflows that require thousands of LLM calls throughout a campaign by models of varying sizes quickly become the new norm. Evidently, using trillion-parameter foundation models for every role in these multi-agent campaigns is prohibitively costly. Small language models like those that we augment in this dissertation to perform commensurately with frontier models emerge as promising workhorses of the imminent agentic future.

Indexing the full body of published science is no longer aspirational. The compute required to scale HiPerRAG from 3.6 million to 200 million papers is approximately 55× the present workload, a target well within the capacity of Aurora at ALCF (note that Aurora was under construction at the time this research was conducted), Frontier at OLCF, and the upcoming generation of exascale systems like Equinox and Solstice pending arrival at Argonne National Laboratory. The question is no longer whether a unified, queryable index over all of published science is technically feasible, but whether the scientific community will commit to building it. This gargantuan undertaking would be best motivated by the realization that, in the near future, humans will interface with the *existing* science through AI agents which will *perceive* the realm of scientific corpus via fast and accurate retrieval.

Retrieval is becoming an interpretive process. The progression from vector-based RAG to reasoning-trace retrieval and ultimately to Swarm Retrieval reflects a fundamental shift: retrieval systems are no longer passive selectors of context, but agents capable of explaining their own decisions. In scientific applications where trust, reproducibility, and auditability are non-negotiable, this shift is not cosmetic, but essential. A system that provides explicit justification for every inclusion and exclusion in the retrieved context not only enables more informed human scrutiny of the final output, but also improves the reasoning trajectories of downstream LLM orchestrators that rely on retrieval as a tool.

6.4 Future Directions

Several research directions follow naturally from this work. *Project ExaForge* [50], currently underway on Aurora, scales reasoning-trace generation to 100 million traces from 10 million papers spanning the entirety of arXiv. The central question driving this endeavor is whether distillation through retrieval can yield general-purpose scientific reasoning capabilities in small language models when the index spans diverse scientific domains. The empirical validation of Swarm Retrieval– beginning with a pure-swarm run on a curated subset and progressing to hybrid evaluations against dense retrieval, late-interaction retrieval, and LLM reranking baselines– is a natural next step. Integration of the resulting retrieval systems with multi-agent scientific discovery frameworks, in which retrieval-augmented small models serve as domain-specialized agents in swarms numbering in the thousands, would test whether the contributions of this dissertation can support not only scientific question answering but autonomous scientific discovery itself.

REFERENCES

[1] PDFMiner. URL https://github.com/pdfminer/pdfminer.six.

[2] PyMuPDF. URL https://pymupdf.readthedocs.io/.

[3] Pypdf. URL https://github.com/py-pdf/pypdf.

[4] Pranav Agarwal et al. ScholarQABench: Evaluating long-form scientific question answering. *arXiv preprint arXiv:2407.11457*, 2024.

[5] Zeeshan Ahmed and Thomas Dandekar. MSL: Facilitating automatic and physical analysis of published scientific literature in PDF format. *F1000Research*, 4:1453, 2017. doi:10.12688/f1000research.7329.2.

[6] Allen Institute for AI. Semantic scholar. https://www.semanticscholar.org/, 2015. Accessed: 2025-08-22.

[7] American Society for Radiation Oncology. RADIATION AND CANCER BIOLOGY STUDY GUIDE, 2023. URL https://www.astro.org/ASTRO/media/ASTRO/AffiliatePages/arro/PDFs/RadBio_StudyGuide_23.pdf.

[8] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. *arXiv preprint arXiv:2310.11511*, 2023.

[9] Akari Asai, Jacqueline He*, Rulin Shao*, Weijia Shi, Amanpreet Singh, Joseph Chee Chang, Kyle Lo, Luca Soldaini, Sergey Feldman, Tian, D'arcy Mike, David Wadden, Matt Latzke, Minyang, Pan Ji, Shengyan Liu, Hao Tong, Bohao Wu, Yanyu Xiong, Luke Zettlemoyer, Dan Weld, Graham Neubig, Doug Downey, Wen-tau Yih, Pang Wei Koh, and Hannaneh Hajishirzi. OpenScholar: Synthesizing scientific literature with retrieval-augmented language models. *Arxiv*, 2024.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[11] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel S. Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin M. Wozniak, Ian Foster, Michael Wilde, and Kyle Chard. Parsl: Pervasive parallel programming in Python. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pages 25–36, 2019.

[12] Philip Bachman, R Devon Hjelm, and William Buchwalter. *Learning representations by maximizing mutual information across views*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[13] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023. URL `https://arxiv.org/abs/2309.16609`.

[14] J. Beattie, S. Neufeld, D.X. Yang, C. Chukwuma, N.B. Desai, M. Dohopolski, and S.B. Jiang. Using large language models to create patient centered consent forms. *International Journal of Radiation Oncology*Biology*Physics*, 120(2):e612, October 2024. ISSN 0360-3016. doi:10.1016/j.ijrobp.2024.07.1346. URL `http://dx.doi.org/10.1016/j.ijrobp.2024.07.1346`.

[15] Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. Small language models are the future of agentic ai, 2025. URL `https://arxiv.org/abs/2506.02153`.

[16] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *Journal of Machine Learning Research*, 2013.

[17] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[18] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.

[19] Daniil A Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624(7992):570–578, 2023.

[20] E. Bolton, D. Hall, M. Yasunaga, T. Lee, C. Manning, and P. Liang. Stanford crfm introduces pubmedgpt 2.7b. Stanford HAI, 2022. URL `https://hai.stanford.edu/news/stanford-crfm-introduces-pubmedgpt-27b`.

[21] Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2022.

[22] Alexander Brace and J. Gregory Pauloski. `https://github.com/braceal/parsl_object_registry`, 2023. Accessed: 2026-04-26.

[23] Garyk Brixi, Matthew G. Durrant, Jerome Ku, Mohsen Naghipourfar, Michael Poli, Gwanggyu Sun, Greg Brockman, Daniel Chang, Alison Fanton, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Jonathan C. Schmok, Ali Taghibakhshi, Anton Vorontsov, Brandon

Yang, Myra Deng, Liv Gorton, Nam Nguyen, Nicholas K. Wang, Michael T. Pearce, Elana Simon, Etowah Adams, Zachary J. Amador, Euan A. Ashley, Stephen A. Baccus, Haoyu Dai, Steven Dillmann, Stefano Ermon, Daniel Guo, Michael H. Herschl, Rajesh Ilango, Ken Janik, Amy X. Lu, Reshma Mehta, Mohammad R. K. Mofrad, Madelena Y. Ng, Jaspreet Pannu, Christopher Ré, John St. John, Jeremy Sullivan, Joseph Tey, Ben Viggiano, Kevin Zhu, Greg Zynda, Daniel Balsam, Patrick Collison, Anthony B. Costa, Tina Hernandez-Boussard, Eric Ho, Ming-Yu Liu, Thomas McGrath, Kimberly Powell, Sudarshan Pinglay, Dave P. Burke, Hani Goodarzi, Patrick D. Hsu, and Brian L. Hie. Genome modelling and design across all domains of life with evo 2. *Nature*, March 2026. ISSN 1476-4687. doi:10.1038/s41586-026-10176-5. URL `http://dx.doi.org/10.1038/s41586-026-10176-5`.

[24] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[25] Harrison Chase. Langchain: Building applications with llms through composability, 2023. URL `https://github.com/langchain-ai/langchain`. Accessed: 2026-04-25.

[26] Bo Chen, Xingyi Cheng, Yangli-ao Geng, Shen Li, Xin Zeng, Boyan Wang, Jing Gong, Chiming Liu, Aohan Zeng, Yuxiao Dong, Jie Tang, and Le Song. xtrimopglm: Unified 100b-scale pre-trained transformer for deciphering the language of protein. *bioRxiv*, 2023. doi:10.1101/2023.07.05.547496. URL `https://www.biorxiv.org/content/10.1101/2023.07.05.547496`.

[27] Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. Dense X retrieval: What retrieval granularity should we use? *arXiv preprint arXiv:2312.06648*, 2023.

[28] Johan SG Chu and James A Evans. Slowed canonical progress in large fields of science. *Proceedings of the National Academy of Sciences*, 118(41):e2021636118, 2021.

[29] Yung-Sung Chuang et al. Dola: Decoding by contrasting layers improves factuality in large language models. *arXiv preprint arXiv:2309.03883*, 2023.

[30] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel Weld. SPECTER: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

[31] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[33] Datalab. Marker: A fast, high fidelity PDF to Markdown converter, 2024. URL https://github.com/VikParuchuri/marker.

[34] DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[35] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

[36] Pritam Deka, Anna Jurek-Loughrey, and P Deepak. Improved methods to aid unsupervised evidence-based fact checking for online health news. *Journal of Data Intelligence*, 3(4):474–504, 2022.

[37] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, 2024.

[38] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[40] Peng Ding. argo-proxy. https://github.com/Oaklight/argo-proxy, 2024. Accessed: 2025-08-23.

[41] Matthijs Douze et al. The Faiss library, 2024.

[42] Yufeng Du et al. Context length alone hurts LLM performance despite perfect retrieval. *arXiv preprint arXiv:2510.05381*, 2025.

[43] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[44] Wenqi Fan, Jiatong Zhao, Xubin Wen, et al. Computational protein science in the era of large language models (LLMs). *arXiv preprint arXiv:2501.10282*, 2025. URL https://arxiv.org/abs/2501.10282.

[45] Daniele Fanelli. Do pressures to publish increase scientists' bias? *PLoS ONE*, 5(4): e10271, 2010. doi:10.1371/journal.pone.0010271.

[46] Hui Feng, Francesco Ronzano, Jude LaFleur, Matthew Garber, Rodrigo de Oliveira, et al. Evaluation of large language model performance on the biomedical language understanding and reasoning benchmark. *medRxiv*, 2024. doi:10.1101/2024.05.17.24307411.

[47] Leo Gao et al. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[48] Tomas Geffner, Kieran Didi, Zuobai Zhang, Danny Reidenbach, Zhonglin Cao, Jason Yim, Mario Geiger, Christian Dallago, Emine Kucukbenli, Arash Vahdat, and Karsten Kreis. Proteina: Scaling flow-based protein structure generative models, 2025. URL `https://arxiv.org/abs/2503.00710`.

[49] Gemini Team, Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[50] Ozan Gökdemir. Exaforge: Distributed llm inference framework for exascale systems, 2026. URL `https://github.com/ogkdmr/ExaForge`. GitHub repository, accessed April 26, 2026.

[51] Ozan Gokdemir, Neil Getty, Robert Underwood, Sandeep Madireddy, Franck Cappello, Arvind Ramanathan, Ian T. Foster, and Rick L. Stevens. Automated mcqa benchmarking at scale: Evaluating reasoning traces as retrieval sources for domain adaptation of small language models. In *Proceedings of the SC '25 Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC Workshops '25, page 545–552, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400718717. doi:10.1145/3731599.3767410. URL `https://doi.org/10.1145/3731599.3767410`.

[52] Ozan Gokdemir, Carlo Siebenschuh, Alexander Brace, Azton Wells, Brian Hsu, Kyle Hippe, Priyanka V. Setty, Aswathy Ajith, J. Gregory Pauloski, Varuni Sastry, Sam Foreman, Huihuo Zheng, Heng Ma, Bharat Kale, Nicholas Chia, Thomas Gibbs, Michael E. Papka, Thomas Brettin, Francis J. Alexander, Anima Anandkumar, Ian Foster, Rick Stevens, Venkatram Vishwanath, and Arvind Ramanathan. HiPerRAG: High-performance retrieval augmented generation for scientific insights. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '25)*, pages 1–13. ACM, 2025. doi:10.1145/3732775.3733586.

[53] Ian J. Goodfellow, Mehdi Mirza, Xia Da, Aaron C. Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *International Conference on Learning Representations*, 2014.

[54] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, et al. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2025. URL `https://arxiv.org/abs/2502.18864`.

[55] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle

Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad

Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

[56] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining for biomedical natural language processing, 2022.

[57] Bernal Jiménez Gutierrez, Yiheng Zhu, Zhonghao Huang, Niklas Kamradt, and Huan Sun. HippoRAG: Neurobiologically inspired long-term memory for large language models. *arXiv preprint arXiv:2405.14831*, 2024.

[58] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

[59] Mark Hanson, Pablo Gomez Barreiro, Paolo Crosetto, and Dan Brockington. The strain on scientific publishing. 2023. doi:10.48550/arXiv.2309.15884.

[60] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025. doi:10.1126/science.ads0018.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for

image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[62] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2021.

[63] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[64] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*, 2023.

[65] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[66] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=nZeVKeeFYf9`.

[67] Lei Huang et al. A survey on hallucination in large language models, 2023.

[68] HuggingFaceTB. Smollm3-3b. `https://huggingface.co/HuggingFaceTB/SmolLM3-3B`, 2025. Accessed: 2025-08-23.

[69] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[70] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL `https://arxiv.org/abs/2310.06825`.

[71] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL `https://arxiv.org/abs/2401.04088`.

[72] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023.

[73] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2020.

[74] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

[75] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021.

[76] Alok Kamatar, J. Gregory Pauloski, Yadu Babuji, Ryan Chard, Mansi Sakarvadia, Daniel Babnigg, Kyle Chard, and Ian Foster. Empowering scientific workflows with federated agents, 2026. URL `https://arxiv.org/abs/2505.05428`.

[77] Greg Kamradt. Semantic chunking, 2023. URL `https://github.com/FullStackRetrieval-com/RetrievalTutorials`.

[78] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[79] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering, 2020.

[80] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference*, pages 39–48, 2020.

[81] Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025. URL `https://arxiv.org/abs/2507.20534`.

[82] James Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114, 2017. doi:10.1073/pnas.1611835114.

[83] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. *arXiv preprint arXiv:2309.06180*, 2023.

[84] Jakub Lála, Odhran O'Donoghue, Alexandr Beránek, Samuel Kluber, Robert Quinton, and Andrew D. White. PaperQA: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*, 2023.

[85] Esther Landhuis. Scientific literature: Information overload. *Nature*, 535:457–458, 2016. doi:10.1038/nj7612-457a.

[86] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.

[87] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[88] Changmao Li and Jeffrey Flanigan. Task contamination: Language models may not be few-shot anymore. *arXiv preprint arXiv:2312.16337*, 2024.

[89] Shen Li et al. PyTorch distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12), 2020.

[90] Zehan Li et al. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

[91] Chen Ling et al. Domain specialization as the key to make large language models disruptive: A comprehensive survey, 2024.

[92] Hongwei Liu et al. ATLAS: A high-difficulty, multidisciplinary benchmark for frontier scientific reasoning. *arXiv preprint arXiv:2511.14366*, 2025.

[93] Jerry Liu. Llamaindex: A data framework for llm applications, 2023. URL `https://github.com/run-llm/llama_index`. Accessed: 2026-04-25.

[94] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi:10.1162/tacl_a_00638.

[95] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. In *Transactions of the Association for Computational Linguistics (TACL)*, volume 8, pages 726–742, 2020.

[96] Yuhan Liu, Yihua Cheng, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Rui Zhang, Kuntai Du, and Junchen Jiang. Lmcache: An efficient

kv cache layer for enterprise-scale llm inference, 2025. URL https://arxiv.org/abs/2510.09665.

[97] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

[98] Nikolaos Livathinos, Christoph Auer, Maksym Lysak, Ahmed Nassar, Michele Dolfi, Panos Vagenas, Cesar Berrospi Ramis, Matteo Omenetti, Kasper Dinkla, Yusik Kim, Shubham Gupta, Rafael Teixeira de Lima, Valéry Weber, Lucas Morin, Ingmar Meijer, Viktor Kuropiatnyk, and Peter W. J. Staar. Docling: An efficient open-source toolkit for AI-driven document conversion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. URL https://arxiv.org/abs/2501.17887.

[99] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024. URL https://arxiv.org/abs/2408.06292.

[100] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. Towards end-to-end automation of AI research. *Nature*, 651:914–919, 2026. doi:10.1038/s41586-026-10265-5.

[101] Renqian Luo, Liang Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. BioGPT: Generative pre-trained transformer for biomedical text generation and mining. *Briefings in Bioinformatics*, 23(6), 2022.

[102] Yizhen Luo, Kai Yang, Massimo Hong, Xing Yi Liu, and Zaiqing Nie. BioMedGPT: Open multimodal generative pre-trained transformer for biomedicine. *arXiv preprint arXiv:2308.09442*, 2023.

[103] Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, Lun Zhou, and Xiaohui Yan. Evoscientist: Towards multi-agent evolving AI scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127*, 2026.

[104] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*, 2023.

[105] Lokman I Meho. The rise and rise of citation analysis. *Physics World*, 20(1):32–36, 2007. doi:10.1088/2058-7058/20/1/33.

[106] Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, 2025. Accessed: April 2025.

[107] Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. *arXiv preprint arXiv:2305.14251*, 2023.

[108] Ludovico Mitchener, Samuel G. Rodriques, et al. Kosmos: An AI Scientist for autonomous discovery. *arXiv preprint arXiv:2511.02824*, 2025.

[109] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agrawal, Xuxi Chen, Anastasia Raber, Erik Jones, Kriti Krber, et al. Orca 2: Teaching small language models how to reason. *arXiv preprint arXiv:2311.11045*, 2023.

[110] Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2023.

[111] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agrawal, Hamid Palangi, and Ahmed Awadallah. Orca: Progressive learning from complex explanation traces of GPT-4. *arXiv preprint arXiv:2306.02707*, 2023.

[112] Deepak Narayanan et al. PipeDream: Generalized pipeline parallelism for DNN training. *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, pages 1–15, 2019.

[113] National Center for Biotechnology Information (NCBI). Pubmed. `https://pubmed.ncbi.nlm.nih.gov/`, 2026. Accessed: 2026-04-13.

[114] Ushma S. Neill. Publish or perish, but at what cost? *Journal of Clinical Investigation*, 118(7):2368, 2008. doi:10.1172/jci36371.

[115] Mathias Wullum Nielsen and Jens Peter Andersen. Global citation inequality is on the rise. *Proceedings of the National Academy of Sciences*, 118(7):e2012208118, 2021. doi:10.1073/pnas.2012208118.

[116] Joel Niklaus, Guilherme Penedo, Hynek Kydlicek, Elie Bakouch, Lewis Tunstall, Ed Beeching, Thibaud Frere, Colin Raffel, Leandro von Werra, and Thomas Wolf. The synthetic data playbook: Generating trillions of the finest tokens, 2026.

[117] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. In *arXiv preprint arXiv:1901.04085*, 2019.

[118] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[119] OpenAI. Learning to reason with LLMs. *OpenAI Blog*, 2024. URL `https://openai.com/index/learning-to-reason-with-llms/`.

[120] OpenAI. Introducing gpt-5, 2025. URL https://openai.com/index/introducing-gpt-5/.

[121] Linke Ouyang, Yuan Qu, Hongbin Zhou, Jiawei Zhu, Rui Zhang, Qunshu Lin, Bin Wang, Zhiyuan Zhao, Man Jiang, Xiaomeng Zhao, Jin Shi, Fan Wu, Pei Chu, Minghao Liu, Zhenxiang Li, Chao Xu, Bo Zhang, Botian Shi, Zhongying Tu, and Conghui He. OmniDocBench: Benchmarking diverse PDF document parsing with comprehensive annotations, 2024. URL https://arxiv.org/abs/2412.07626. Accepted at CVPR 2025.

[122] Long Ouyang et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[123] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[124] Long Phan et al. Humanity's last exam. *arXiv preprint arXiv:2501.14249*, 2025.

[125] Aurora Pi-Groeneveld, Luca De Martini, Jesse Dodge, Oyvind Tafjord, Ben Hutchinson, Matt Gardner, Noah Smith, Luke Zettlemoyer, and Nathan Schneider. Olmo: Accelerating the science of language models. 2024.

[126] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free, 2025. URL https://arxiv.org/abs/2505.06708.

[127] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[128] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

[129] Sara Reardon. 'elite' researchers dominate citation space. *Nature*, 591(7849):333–334, 2021. doi:10.1038/d41586-021-00553-7.

[130] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[131] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[132] David Rein et al. GPQA: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[133] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[134] Shafiq Rayhan Joty Caiming Xiong Yingbo Zhou Semih Yavuz Rui Meng, Ye Liu. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog, 2024. URL `https://www.salesforce.com/blog/sfr-embedding/`.

[135] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[136] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*, 2022.

[137] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

[138] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2020.

[139] Maulik Shukla et al. BV-BRC: A unified bacterial and viral bioinformatics resource with expanded functionality and AI integration. *Nucleic Acids Research*, 2025. doi:10.1093/nar/gkaf1254.

[140] Carlo Siebenschuh, Kyle Hippe, Ozan Gokdemir, Alexander Brace, Arham Khan, Khalid Hossain, Yadu Babuji, Nicholas Chia, Venkatram Vishwanath, Rick Stevens, Arvind Ramanathan, Ian Foster, and Robert Underwood. AdaParse: An adaptive parallel PDF parsing and resource scaling engine. In *Eighth Conference on Machine Learning and Systems (MLSys)*, 2025.

[141] Matthew Sinclair et al. Scalable agentic reasoning for designing biologics targeting intrinsically disordered proteins. *arXiv preprint arXiv:2512.15930*, 2025.

[142] Aditi Singh, Abul Ehtesham, Gaurav Kumar, and Chandrani Saxena. Agentic retrieval-augmented generation: A survey on agentic RAG. *arXiv preprint arXiv:2501.09136*, 2025.

[143] Amanpreet Singh, Mike D'Arcy, Arman Cohan, Doug Downey, and Sergey Feldman. SciRepEval: A multi-format benchmark for scientific document representations. *arXiv preprint arXiv:2211.13308*, 2023.

[144] Karan Singhal, Shekoofeh Azizi, Tao Tu, et al. Large language models encode clinical knowledge. *Nature*, 620:172–180, 2023.

[145] Michael Skarlinski et al. PaperQA2: A science agent for high quality, comprehensive literature search and question answering. *arXiv preprint arXiv:2409.13740*, 2024.

[146] Luca Soldaini and Kyle Lo. peS2o: Pretraining efficiently on S2ORC. `https://github.com/allenai/peS2o`, 2023. Allen Institute for AI.

[147] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi:10.18653/v1/2024.acl-long.840. URL `https://aclanthology.org/2024.acl-long.840`.

[148] Luca Soldaini et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. *arXiv preprint arXiv:2402.00159*, 2024.

[149] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[150] Ross Taylor et al. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

[151] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric

Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL `https://arxiv.org/abs/2503.19786`.

[152] SGLang Team. Sglang: Efficient execution of structured generation programs. `https://github.com/sgl-project/sglang`, 2025.

[153] Yuan-Sen Ting, Tuan Dung Nguyen, Tirthankar Ghosal, Rui Pan, Hardik Arora, Zechang Sun, Tijmen de Haan, Nesar Ramachandra, Azton Wells, Sandeep Madireddy, and Alberto Accomazzi. Astromlab 1: Who wins astronomy jeopardy!?, 2024. URL `https://arxiv.org/abs/2407.11194`.

[154] Together Computer. RedPajama: an open dataset for training large language models, 2023. URL `https://github.com/togethercomputer/RedPajama-Data`.

[155] George Tsatsaronis et al. An overview of the BioASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138, 2015.

[156] Ultralytics. YOLOv5: A state-of-the-art real-time object detection system. `https://docs.ultralytics.com`, 2021. April 2026.

[157] Richard Van Noorden. Scientists may be reaching a peak in reading habits. *Nature*, 2014. doi:10.1038/nature.2014.14658.

[158] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[159] David Wadden et al. Fact or fiction: Verifying scientific claims, 2020.

[160] Bin Wang, Chao Xu, Xiaomeng Zhao, Linke Ouyang, Fan Wu, Zhiyuan Zhao, Rui Xu, Kaiwen Liu, Yuan Qu, Fukai Shang, Bo Zhang, Liqun Wei, Zhihao Sui, Wei Li, Botian Shi, Yu Qiao, Dahua Lin, and Conghui He. MinerU: An open-source solution for precise document content extraction. 2024. URL `https://arxiv.org/abs/2409.18839`.

[161] Feng Wang and Huaping Liu. Understanding the behaviour of contrastive loss. *arXiv preprint arXiv:2012.09740*, 2021.

[162] Liang Wang et al. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2024.

[163] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2023.

[164] Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2526–2547, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi:10.18653/v1/2025.acl-long.127. URL `https://aclanthology.org/2025.acl-long.127/`.

[165] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2022.

[166] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022.

[167] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.

[168] Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

[169] Orion Weller, Michael Boratko, Iftekhar Naim, and Jinhyuk Lee. On the theoretical limitations of embedding-based retrieval, 2026. URL https://arxiv.org/abs/2508.21038.

[170] Yijun Xiao and William Yang Wang. On hallucination and predictive uncertainty in conditional language generation, 2021.

[171] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2021.

[172] Yan Xu et al. KILM: Knowledge injection into encoder-decoder language models, 2023.

[173] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

[174] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2023.

[175] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[176] Tianzhu Ye, Li Dong, Zewen Chi, Xun Wu, Shaohan Huang, and Furu Wei. Black-box on-policy distillation of large language models. *arXiv preprint arXiv:2511.10643*, November 2025.

[177] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, et al. GLM-5: From vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026. URL https://arxiv.org/abs/2602.15763.

[178] Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. Tinyllama: An open-source small language model, 2024. URL https://arxiv.org/abs/2401.02385.

[179] Shengyu Zhang et al. Instruction tuning for large language models: A survey, 2024.

[180] Wangmeng Zuo, Faqiang Wang, David Zhang, Liang Lin, Yuchi Huang, Deyu Meng, and Lei Zhang. Iterated support vector machines for distance metric learning. *IEEE Transactions on Cybernetics*, 45(7):1257–1268, 2015.

[181] Maxim Zvyagin, Alexander Brace, Kyle Hippe, Yuntian Deng, Bin Zhang, et al. GenSLMs: Genome-scale language models reveal SARS-CoV-2 evolutionary dynamics. *The International Journal of High Performance Computing Applications*, 37(6): 683–705, 2023.