PARAKG: SCALABLE CORPUS-LEVEL KNOWLEDGE GRAPH CONSTRUCTION FROM SCIENTIFIC DOCUMENTS WITH PARALLEL LLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graphs (KGs) are essential for organizing large collections of unstructured scientific literature. While existing LLM-based methods can effectively construct local KGs from individual documents, scaling to corpus-level graphs poses both severe computational bottlenecks and cross-document integration challenges. To address these challenges, we propose **ParaKG**, a parallel multi-agent framework for large-scale scientific KG construction. ParaKG distributes the document extraction workload across dynamically scheduled agents and merges the resulting local graphs via a hierarchical binary-tree reduction. To scale entity resolution without sacrificing precision, we introduce a two-tier mechanism combining fast lexical filtering with context-grounded LLM verification and persistent caching. We evaluate ParaKG on real-world materials science corpora, demonstrating linear speedups during document-level extraction. Our analysis also characterizes sub-linear scaling bottlenecks during the global merge phase, investigates non-deterministic concurrency effects on graph topology, and evaluates stage-specific performance trade-offs across diverse large language model architectures.

1 INTRODUCTION

Scientific knowledge is increasingly distributed across large collections of literature. Knowledge graphs (KGs) provide a foundational representation to organize this unstructured text into structured, queryable entities and relations, empowering critical applications such as automated hypothesis generation (Wang et al., 2024), retrieval-augmented generation (RAG) (Edge et al., 2024; He et al., 2024; Li et al., 2026b; Zhu et al., 2026), and literature-scale semantic search (Tiwari et al., 2025; Sun et al., 2025b).

While LLMs have made single-document KG extraction increasingly practical (Zhang & Soh, 2024; Bai et al., 2025; Mo et al., 2025), constructing a unified KG across large scientific corpora presents two fundamental challenges. First, *computational scalability*: processing thousands of papers requires tens of thousands of LLM inference calls, making naive sequential execution prohibitively slow and naive parallelization prone to scheduling bottlenecks. Second, *cross-document semantic integration*: isolated document-level graphs exhibit conflicting terminology, polysemy, and abbreviation mismatches that require scalable entity resolution without incurring quadratic pairwise comparison costs.

To address these challenges, we propose *ParaKG*, a parallel multi-agent framework for large-scale scientific KG construction. ParaKG operates in three stages. First, each document is parsed and partitioned into discrete text chunks, which are dispatched as parallel extraction tasks to available LLM agents. Second, each agent independently extracts entities and relations from its assigned chunks using structured LLM prompting. Third, the extracted elements are merged through a hierarchical binary-tree reduction. At each level of the tree, pairs of sub-graphs are aligned using our two-tier entity resolution mechanism, which combines fast lexical and abbreviation filtering with context-grounded LLM verification backed by persistent lookup caches, before being consolidated into a unified graph.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We evaluate ParaKG on two materials science corpora, analyzing parallel efficiency, graph topological integrity, and cross-model runtimes. Our experiments show that while document-level extraction scales linearly with additional agents, the global merge phase scales sub-linearly and dominates end-to-end runtime at higher concurrency. We further observe that greater parallelism introduces slight non-deterministic variance in the resulting graph topology, and that different LLM architectures exhibit distinct stage-specific bottlenecks, revealing key trade-offs among concurrency, structural consistency, and model choice.

The main contributions of this paper are as follows:

- We formulate large-scale text-to-KG construction as a corpus-level graph integration problem and introduce *ParaKG*, a parallel multi-agent framework designed to produce a unified knowledge graph from heterogeneous scientific literature.
- We design a scalable architecture that comprises a chunk-level parallel extraction scheduler with dynamic request batching and a hierarchical binary-tree reduction mechanism with two-tier entity resolution.
- We evaluate ParaKG on real-world materials science corpora, characterizing scaling behaviors, cross-model bottlenecks, and concurrency effects on graph topology.

2 Related Work

2.1 Knowledge Graph Construction

Conventional KG construction relies on rigid pipelines of ontology engineering, rule-based extraction, and manual fusion, which often struggle with scalability and cross-domain generalization (Bian, 2025). Large Language Models (LLMs) overcome these bottlenecks by enabling generative knowledge modeling and semantic unification directly from natural language. Recent efforts generally fall into two paradigms: top-down schema modeling and bottom-up extraction. Top-down approaches use LLMs to translate competency questions into formal ontologies (Lippolis et al., 2025; Saeedizade & Blomqvist, 2024) or induce conceptual hierarchies directly from scientific corpora (Fathallah et al., 2025; 2024; Sun et al., 2025a). While effective for domain modeling, these methods often require significant manual guidance and are difficult to scale across massive, heterogeneous document collections.

In contrast, bottom-up extraction builds dynamic graph structures directly from raw text to provide LLMs with structured memory and factual grounding. Systems like GraphRAG (Edge et al., 2024) and OntoRAG (Tiwari et al., 2025) demonstrate how extracting entities and community structures can support global corpus reasoning. To handle schema evolution, recent pipelines increasingly adopt hybrid approaches: open information extraction produces initial triples (e.g., using two-stage generation to reduce cognitive load (Mo et al., 2025)), followed by semantic clustering and schema normalization (Zhang & Soh, 2024; Ye et al., 2023; Bai et al., 2025). Our work aligns with this bottom-up paradigm but focuses specifically on the systems challenge of parallelizing this extraction and unification process across large document clusters using a coordinated multi-agent architecture.

2.2 Graph-Assisted Large Language Models

A complementary line of work studies how graph structures can assist LLMs after or alongside model training. Luo et al. organize graph-assisted LLMs around the limitations of LLM knowledge, reasoning, and collaboration (Luo et al., 2026). For knowledge augmentation, graphs can either modify model parameters through knowledge injection, editing, or unlearning, or remain external through graph-assisted retrieval and memory. This distinction is important for KG construction because our setting emphasizes non-parametric grounding: the graph is built and queried as an external structure rather than absorbed into the model weights. The survey further highlights graph-assisted validation and correction, where generated claims are converted into graph-structured representations and checked against trusted KGs or internal dependency graphs. This perspective connects directly to construction quality control, since extracted entities and relations can be validated through graph consistency, provenance, and cross-document agreement. Beyond knowledge, graph-assisted reasoning and planning represent intermediate thoughts, tool dependencies, or action flows as explicit graph structures, enabling revisitable and branching

inference rather than purely linear chain-of-thought generation. Finally, graph-assisted collaboration models experts or agents as nodes and communication dependencies as edges, providing a natural abstraction for coordinating parallel LLM workers. These themes motivate our use of KGs not only as final artifacts, but also as coordination and evaluation structures for scalable document understanding.

2.3 Multi-agent Systems

LLM-based multi-agent systems extend single-agent workflows by decomposing complex tasks across specialized agents, but recent work shows that coordination architecture strongly determines latency, adaptability, and reliability. Parallelism Meets Adaptiveness proposes adaptive coordination for document understanding through dynamic task routing, bidirectional feedback, and parallel agent evaluation, showing how structured competition and feedback can improve multi-agent reasoning over long, heterogeneous documents (Xia et al., 2025). M1-Parallel addresses the latency of sequential multi-step agent workflows by running multiple multi-agent teams concurrently and aggregating or terminating trajectories based on progress, framing parallel execution as a way to trade additional test-time compute for lower wall-clock latency or higher completion rates (Zhang et al., 2025). In embodied settings, Parallelized Planning-Acting decouples planning and acting into concurrent, interruptible threads for Minecraft agents, reducing the bottleneck caused by serialized deliberation before action (Li et al., 2025). More broadly, General AgentBench studies sequential and parallel test-time scaling for general LLM agents and highlights that parallel sampling alone creates a verifiability gap when agents cannot reliably select the best trajectory from multiple candidates (Li et al., 2026a). For knowledge-intensive applications, AGENTiGraph connects multi-agent orchestration directly to KG construction and interaction: specialized agents perform intent classification, task planning, and automatic knowledge integration so users can incrementally build, query, and refine domain-specific KGs through natural language (Zhao et al., 2025). Together, these systems suggest that parallel agents are most useful when coupled with explicit coordination, evaluation, and grounding structures rather than treated as independent repeated samples.

3 The ParaKG Framework

In this section, we describe our multi-agent ParaKG framework for constructing scientific knowledge graphs from large collections of scientific literature. As shown in Figure 1, the framework consists of three main components: document processing, distributed information extraction, and cross-document entity resolution. This design enables high-throughput processing while integrating isolated knowledge graphs from individual papers into an interconnected knowledge network.

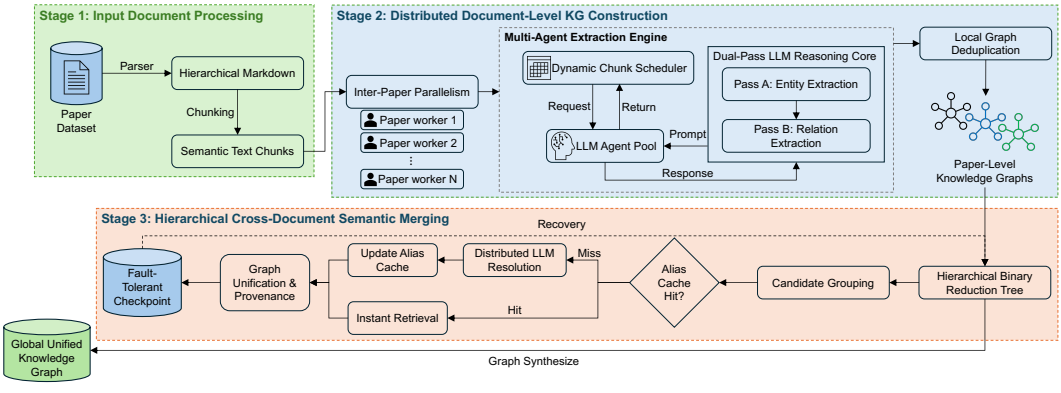


Figure 1: Overview of the ParaKG framework for constructing scientific knowledge graphs.

3.1 Document Processing

To maximize processing throughput, the pipeline begins by using a lightweight, high-speed text extractor to convert raw PDFs into Markdown format. While this rapid extraction inherently retains some formatting noise and inline artifacts, we intentionally shift the burden of semantic regularization downstream to the LLM agents, which are highly robust at cleaning and recovering

accurate terminology (e.g., chemical formulas) from noisy inputs. The raw text is then partitioned into discrete chunks subject to a fixed character limit θ_{chunk}. Boundaries are strictly aligned to sentence endings to prevent mid-sentence splits, preserving local grammatical coherence while respecting model context-window limits.

3.2 Distributed Information Extraction

Extracting structured scientific facts from text chunks requires balancing ontological precision with high-throughput computational scalability. The extraction process operates under domain-tailored contextual constraints. In our case, it is specifically tuned for materials science, emphasizing core entities such as crystal phases, synthesis routes, computational descriptors, and electrochemical performance metrics. To capture methodological insights from background text and facilitate novel idea generation (Wang et al., 2024), the extraction prompt explicitly formalizes functional dependencies into directed relationships indicating that a given algorithm, characterization technique, or material strategy is employed for a concrete scientific problem, prediction task, or degradation analysis. Furthermore, the extraction rules explicitly prevent the model from merging distinct chemical formulas, base materials with their doped variants, or general material classes with their specific compounds.

To execute this extraction at scale, ParaKG exploits two levels of parallelism, as illustrated in Algorithm 1. At the **inter-document level** (line 17), multiple paper workers process separate documents concurrently via ParallelExecute. At the **chunk level** (lines 7–12), each worker further parallelizes across the semantic chunks of its assigned document: threads acquire an available request slot from a synchronized agent pool under a per-agent request limit (line 8), enabling batched requests to be served concurrently across the available LLM agents (line 1). Each thread runs a two-pass extraction in which Pass A discovers named scientific entities (line 9) and Pass B extracts context-grounded relational triples anchored to those entities (line 10), releasing the request slot upon completion (line 12). The per-chunk results are then consolidated into a single-paper knowledge graph (line 14), which performs fast intra-document deduplication through in-memory lexical normalization and set union, requiring no LLM invocations.

Algorithm 1 Distributed Document-Level KG Extraction

Input: Corpus of scientific documents $\mathcal{D} = \{D_1, \ldots, D_N\}$, LLM agent pool $\mathcal{P}$ of size M with per-agent request limit R, chunk size θ_{chunk}, extraction context $\mathcal{C}$, number of paper workers W.
Output: Set of document-level knowledge graphs $\mathcal{G} = \{G_1, \ldots, G_N\}$.

1: Initialize synchronized agent pool $\mathcal{P}$ with $M \times R$ concurrent request capacity
2: Initialize output graph collection $\mathcal{G} \leftarrow \emptyset$
3: **function** ExtractDocument(D_i)
4: $\quad \mathcal{M} \leftarrow$ ParsePDFToMarkdown(D_i) ▷ Parse PDF and convert to Markdown
5: $\quad \mathcal{S} \leftarrow$ SemanticChunking($\mathcal{M}, \theta_{\text{chunk}}$) ▷ Break down into manageable chunks
6: $\quad$ Initialize entity set $\mathcal{E}_i \leftarrow \emptyset$, relation set $\mathcal{R}_i \leftarrow \emptyset$
$\quad$ ▷ **Dynamic Chunk-Level Parallelism**
7: $\quad$ **for each** chunk $c_j \in \mathcal{S}$ **in parallel do**
8: $\quad\quad a \leftarrow \mathcal{P}$.Acquire() ▷ Acquire request slot under limit R
9: $\quad\quad \mathcal{E}_{ij} \leftarrow a$.ExtractEntities($c_j, \mathcal{C}$) ▷ Pass A: Entity discovery
10: $\quad\quad \mathcal{R}_{ij} \leftarrow a$.ExtractRelations($c_j, \mathcal{E}_{ij}, \mathcal{C}$) ▷ Pass B: Relational binding
11: $\quad\quad$ **atomically** $\mathcal{E}_i \leftarrow \mathcal{E}_i \cup \mathcal{E}_{ij}, \quad \mathcal{R}_i \leftarrow \mathcal{R}_i \cup \mathcal{R}_{ij}$
12: $\quad\quad \mathcal{P}$.Release(a) ▷ Return request slot to pool $\mathcal{P}$
13: $\quad$ **end for**
14: $\quad G_i \leftarrow$ AssembleGraph($\mathcal{E}_i, \mathcal{R}_i$) ▷ Intra-document union and deduplication
15: $\quad$ **return** G_i
16: **end function**
$\quad$ ▷ **Inter-Document Parallelism**
17: $\mathcal{G} \leftarrow$ ParallelExecute(ExtractDocument, $\mathcal{D}, W$)
18: **return** $\mathcal{G}$

3.3 CROSS-DOCUMENT GRAPH INTEGRATION

Consolidating independent single-paper graphs into a unified global knowledge graph requires resolving synonyms, phrasing variations, and checking abbreviations across documents without losing fine-grained physical distinctions. Because different authors use alternative naming conventions for identical materials and properties, the framework must identify equivalent entities while preventing the over-merging of scientifically distinct concepts.

To address this challenge, ParaKG uses a hierarchical binary tree reduction strategy, as detailed in Algorithm 2. Rather than attempting a single global merge, adjacent pairs of graphs are merged in parallel at each round (lines 17–20), reducing the queue size by half until a single unified graph remains. This tree-structured reduction exposes $\lceil \log_2 N \rceil$ rounds of parallelism and avoids the quadratic cost of all-pairs entity comparison, making the approach easily applicable to large corpora.

Algorithm 2 Hierarchical Cross-Document Semantic Merging

Input: Document graphs $\mathcal{G} = \{G_1, \ldots, G_N\}$ with $G_i = (V_i, E_i, \Pi_i)$, Agent pool $\mathcal{P}$, Context $\mathcal{C}$.
Output: Globally unified knowledge graph $G^* = (V^*, E^*, \Pi^*)$.

1: Initialize queue $\mathcal{Q} \leftarrow \mathcal{G}$ and alias cache $\mathcal{M} \leftarrow \{u \mapsto u\}$
2: **function** MERGEGRAPHS(G_A, G_B)
3: **for each** cluster $B \in$ CLUSTERCANDIDATES$(V_A \cup V_B)$ with $|B| > 1$ **do**
4: **if** $\exists u, v \in B : \mathcal{M}(u) \neq \mathcal{M}(v)$ **then** ▷ **Cache miss:** unresolved pair
5: $\mathcal{M}_B \leftarrow \mathcal{P}$.ALIGNENTITIES$(B, \mathcal{C})$ ▷ Acquire agent for LLM alignment
6: **if** $\mathcal{M}_B \neq \emptyset$ **then**
7: **atomically** $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{M}_B$ ▷ Update alias cache
8: **end if**
9: **end if** ▷ **Cache hit:** skip LLM if already resolved
10: **end for**
11: $V_{AB} \leftarrow \{\mathcal{M}(u) \mid u \in V_A \cup V_B\}$ ▷ Retrieve canonical vertices from $\mathcal{M}$
12: $E_{AB} \leftarrow \{(\mathcal{M}(u), r, \mathcal{M}(v)) \mid (u, r, v) \in E_A \cup E_B\}$ ▷ Remap edges via $\mathcal{M}$
13: $\Pi_{AB} \leftarrow \Pi_A \cup \Pi_B$ ▷ Union provenance records
14: **return** $G_{AB} = (V_{AB}, E_{AB}, \Pi_{AB})$
15: **end function**
▷ **Hierarchical binary tree reduction**
16: **while** $|\mathcal{Q}| > 1$ **do**
17: Pairs $\leftarrow \{(\mathcal{Q}_{2k-1},\ \mathcal{Q}_{2k}) \mid 1 \leq k \leq \lfloor |\mathcal{Q}|/2 \rfloor\}$ ▷ Pair adjacent graphs
18: $\mathcal{Q}_{\text{next}} \leftarrow$ PARALLELEXECUTE(MERGEGRAPHS, Pairs, $\mathcal{P}$)
19: $\mathcal{Q} \leftarrow \mathcal{Q}_{\text{next}} \cup (\{\mathcal{Q}_{|\mathcal{Q}|}\}$ if $|\mathcal{Q}|$ is odd else $\emptyset)$ ▷ Carry unpaired graph
20: **end while**
21: **return** $G^* \leftarrow \mathcal{Q}[0]$

The key challenge within each pairwise merge is entity resolution: determining which names across the two graphs refer to the same scientific concept, without over-merging scientifically distinct ones. Exhaustively querying an LLM for all candidate pairs would incur prohibitive computational overhead and latency. To overcome this, ParaKG introduces a two-tier resolution mechanism accelerated by an alias cache, implemented as a direct key-value mapping to prevent unchecked transitive over-merging. To support safe concurrent access across thousands of parallel workers, this global cache is maintained as an in-memory dictionary that is deterministically reconciled and synchronized by the main orchestrator process. In the first tier, a lightweight lexical filter clusters candidate entity names by shared word tokens, abbreviation expansion, and character string similarity. This grouping step is inexpensive and intentionally permissive, narrowing the space of potential synonyms without discarding true semantic matches.

In the second tier, candidate clusters containing names not yet consistently resolved in the alias cache trigger a cache miss. These unresolved clusters are dispatched to LLM agents to evaluate the names in scientific context and select a canonical representative. If valid synonyms are found, the resolved mappings are written back to the alias cache. Once entity alignment is complete, the canonical mappings are applied uniformly to merge entity nodes, remap relational edges, and union per-triple provenance records into the output graph.

4 EVALUATION OF PERFORMANCE

We evaluate the proposed ParaKG framework to address three critical questions for large-scale KG construction: (1) how to optimize system parameters like chunk size and request concurrency to maximize extraction throughput; (2) how efficiently the two-tier architecture scales across multiple LLM agents and varying model architectures; and (3) whether highly parallelized extraction and merging mechanisms preserve the structural integrity and semantic coverage of the final knowledge graph. Unless otherwise stated, all configurations operate on the same materials science corpora using identical extraction schemas and post-processing rules. We utilize single-agent serial execution as our primary baseline to isolate the performance gains, scheduling overheads, and topological stability of our distributed architecture.

4.1 EXPERIMENTAL SETUP

Hardware and Cluster Infrastructure. All experiments are conducted on a high-performance computing (HPC) cluster. Compute nodes feature a 64-core AMD EPYC processor, 512 GB DDR4 memory, and four NVIDIA A100 GPUs (40 GB VRAM each). Thus, deploying our maximum 256-agent configuration requires 64 compute nodes.

Datasets and LLM Serving. We evaluate our framework on two materials science literature corpora spanning different scales and document granularities, as summarized in Table 1. Specifically, BATTERY-173 covers battery materials literature at small scale, while XAS-3.5K targets X-ray Absorption Spectroscopy (XAS) research (He et al., 2026). For LLM serving, we benchmark open-weight models including Gemma 4, Qwen 3.8, and distilled DeepSeek R1. All models have approximately 30 billion parameters and are deployed via VLLM with 4-bit BitsAndBytes quantization and maximum context length set to 8,192 tokens in order to fit one LLM agent per physical GPU.

Dataset	Format	Papers	Average Text Length	Size
BATTERY-173	PDF	173	56.3k chars	558 MB
XAS-3.5K	JSONL	3,505	42.9k chars	146 MB

Table 1: Overview of the materials science literature datasets used in our experiments.

4.2 SYSTEM PARAMETER OPTIMIZATION

To optimize knowledge graph extraction efficiency and throughput, we analyze two pivotal pipeline parameters: the document chunk size (θ_{chunk}) and the maximum requests for each LLM agent (R).

Document Chunk Size (θ_{chunk}). Chunk size introduces a trade-off between extraction throughput and contextual integrity. We evaluate this using 16 Gemma 4 agents on BATTERY-173, with a limit of 4 concurrent requests per agent ($R = 4$) to isolate task scheduling behaviors.

As shown in Figure 2, the average per-paper extraction time follows a U-shaped curve. This trend reflects the trade-off between task granularity and per-chunk processing time. Extreme fragmentation ($\theta_{\text{chunk}} = 1{,}000$) reduces throughput due to fixed API and prompt overheads. Conversely, very large chunks ($\theta_{\text{chunk}} \geq 8{,}000$) pack too much text into a single request, which proportionally extends the LLM's autoregressive generation time. Because each individual extraction task takes much longer to process, the overall pipeline throughput drops even when the scheduler is fully saturated. We observe the highest efficiency at $\theta_{\text{chunk}} = 2{,}000$, which processes documents 39.4% faster than the 1,000-character baseline.

In terms of graph density, the average number of entities and relations extracted per paper decreases as chunk sizes increase from 1,000 to 4,000 characters and then plateaus. This initial drop suggests that excessively small chunks ($\theta_{\text{chunk}} \leq 2{,}000$) artificially inflate the entity count due to severe text fragmentation, forcing the model to extract unresolved aliases across severed contexts. While we lack a gold-standard benchmark to definitively rule out coverage loss, the fact that average graph density stabilizes and remains essentially flat from 4,000 to 10,000 characters implies that 4,000 characters is sufficient to resolve most local coreferences. We therefore use $\theta_{\text{chunk}} = 4{,}000$ as our default operating point. This size limits fragmentation artifacts while still providing a 33.8%

speedup over the baseline. Note that the optimal chunk size might differ at higher concurrency limits such as $R = 24$.

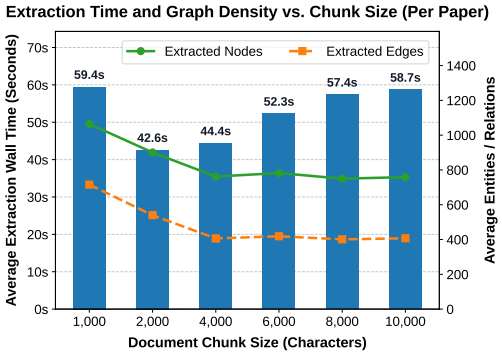


Figure 2: Effect of document chunk size (θ_{chunk}) on extraction efficiency and graph density.

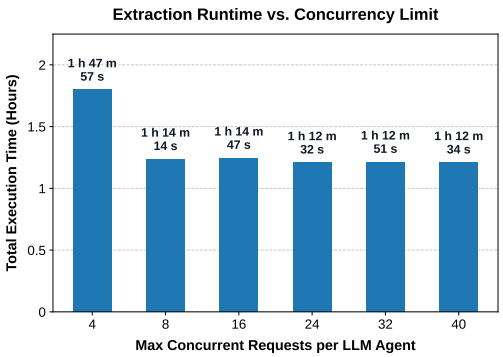


Figure 3: Effect of per-agent request limit (R) on extraction time.

Max Requests per Agent (R). LLM serving engines rely on iteration-level dynamic request batching to maximize hardware throughput. We evaluate concurrency sweeps across $R \in \{4, 8, 16, 24, 32, 40\}$ using 16 agents on BATTERY-173 (see Figure 3). At $R = 4$, the pipeline is severely under-saturated, requiring 1 h 47 m 57 s. Increasing concurrency to $R = 8$ achieves a dramatic 31.2% runtime reduction down to 1 h 14 m 14 s.

Beyond $R = 8$, runtime shows diminishing returns. It changes little at $R = 16$ (1 h 14 m 47 s) and then reaches a plateau for $R \in \{24, 32, 40\}$, where all measurements lie within 20 seconds of one another around 1 h 12 m 40 s. Although the A100 GPUs can support substantially more concurrent requests without exhausting the KV cache, this plateau indicates a client-side upstream bottleneck. Our setup uses $W = 16$ paper workers to distribute documents, which limits the number of chunk-extraction tasks that can be generated concurrently. Once R is sufficiently large, the aggregate capacity of the 16 LLM agents exceeds this task-generation rate, after which higher R values offer no marginal gain. We adopt $R = 24$ (1 h 12 m 32 s) as the default operating point. Increasing the limit to $R = 40$ (1 h 12 m 34 s) produces an effectively identical runtime. We note that scaling the paper workers (W) could unlock significantly higher throughput.

4.3 CROSS-MODEL PERFORMANCE

In the following sections, we measure the end-to-end pipeline runtime of ParaKG. Note that the initial data preprocessing stage, which involves parsing PDFs and segmenting text into chunks, accounts for less than 1% of the total runtime. We therefore consider this overhead computationally trivial and omit it from our runtime analysis. Instead, we primarily focus on the LLM-driven information extraction and semantic merging stages, as they represent the dominant computational bottlenecks of the pipeline.

We compare the end-to-end pipeline runtime across different LLMs deployed in our framework, including Gemma 4, Qwen 3.8, and distilled DeepSeek R1. To ensure a standardized evaluation of overall system throughput across these diverse models, all experiments were conducted under identical conditions using a pool of 16 concurrent agents, a maximum limit of 24 concurrent requests per agent, and a fixed document chunk size of 4,000 characters.

As shown in Figure 4, runtime varies considerably across the evaluated models. Gemma 4 requires approximately 3.3 hours to complete both stages, extracting an average of 695 entities per document. In contrast, Qwen 3.8 requires over 8 hours for the initial extraction phase and yields a 38% lower entity count. DeepSeek R1 extracts slightly more entities than Gemma but requires over 13 hours to complete the semantic merge stage, where the slower throughput is a direct consequence of its reasoning architecture: outputting long chain-of-thought (CoT) sequences for thousands of pairwise merging decisions reduces effective decode speed.

These findings suggest that standard instruction-tuned models like Gemma provide a practical balance of decode speed and extraction density for large-scale KG construction. While reasoning

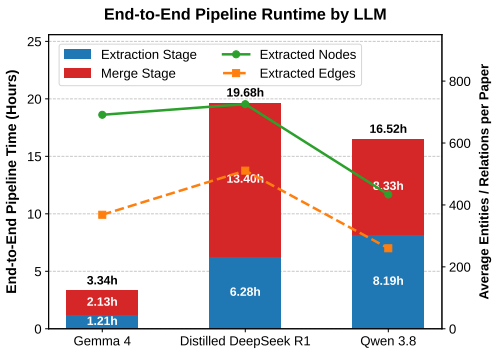


Figure 4: Pipeline runtime comparison across different LLMs.

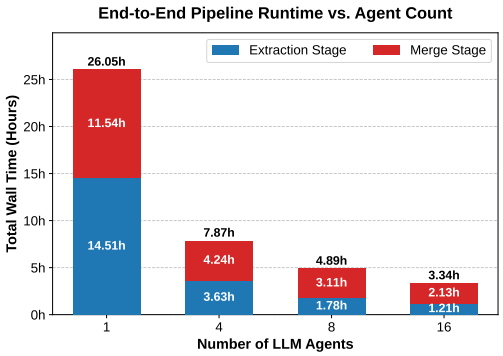


Figure 5: Effect of agent scaling on pipeline runtime, including extraction and merge stages.

models (e.g., DeepSeek R1) may provide better contextual understanding, the overhead of CoT generation makes them too slow for thousands of iterative merging tasks. We therefore use Gemma for all remaining experiments to maintain system throughput, though a rigorous evaluation of extraction precision and recall across these models remains an important area for future work.

4.4 System Scalability

We evaluate the scalability and efficiency of ParaKG by measuring the pipeline runtime across varying agent pool sizes. To demonstrate how the framework accelerates knowledge graph construction on fixed workloads, we scale the number of LLM agents and record the total wall-clock time required to process the Battery-173 dataset using the Gemma 4 model. By tracking the separate time costs of the two primary pipeline stages, document-level extraction and global graph merging, we can identify system bottlenecks and assess how effectively the architecture utilizes additional compute resources.

As shown in Figure 5, the system exhibits different scaling behaviors across the two main stages. The document-level extraction phase demonstrates linear speedups when moving from 1 to 4 agents. However, as the agent count approaches our configured client concurrency limit ($W = 16$ document workers), extraction efficiency drops from 101.9% at 8 agents to 74.9% at 16 agents, confirming that performance is bounded by the client-side scheduling cap rather than LLM serving capacity. In contrast, the semantic merging phase exhibits sub-linear scaling. Because the merging framework utilizes a hierarchical reduction tree, the initial parallel merges efficiently utilize all available agents. However, as the tree approaches the root and the intermediate graphs grow larger, the number of independent merging tasks decreases. This bottleneck restricts maximum concurrency at the top of the tree, limiting the overall speedup of the global merging phase despite the availability of additional agents.

To evaluate the framework on a large-scale corpus, we processed the XAS-3.5K dataset using $W = 256$ paper workers and 256 LLM agents. As detailed in Table 2, this configuration achieved an extraction throughput of 5,498 characters per second, finishing the extraction stage in 7.59 hours. The initial extraction produced a fragmented graph of 1.41 million nodes and 1.31 million edges. The subsequent semantic merge stage, distributed across the same 256 agents, compressed this raw output down to 416.5 K unique nodes and 925.9 K edges. It took 4.84 hours to reduce the node count by over 70%. Compared to the smaller Battery-173 baseline, the 256-agent setup scales effectively, completing the end-to-end construction of a large-scale knowledge graph from thousands of papers in 12.5 hours.

4.5 Deduplication and Structural Stability

We evaluate the deduplication rate and merging variance of the constructed knowledge graph across different execution scales. While the extraction and merging mechanics are designed to be scalable, the use of a globally shared, concurrently populated alias cache introduces non-deterministic ordering effects during the hierarchical reduction process.

Table 2: End-to-end performance comparison across datasets.

	BATTERY-173	XAS-3.5K
Agents	16	256
Extraction Stage		
Time (h)	1.21	7.59
Sec/Paper	25.18	7.80
Chars/s	2,238	5,498
Nodes	119.7 K	1.41 M
Edges	64.1 K	1.31 M
Merge Stage		
Nodes	81.7 K	416.5 K
Edges	62.4 K	925.9 K
Time (h)	2.13	4.84

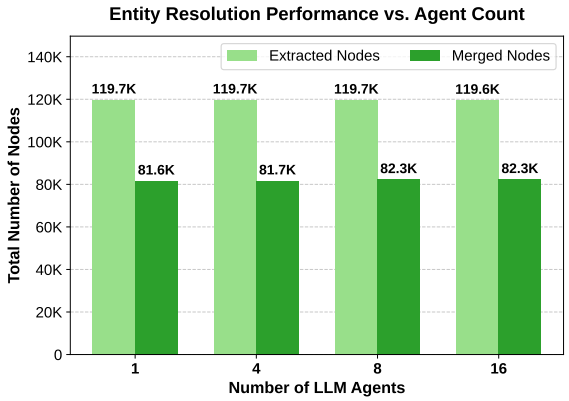


Figure 6: Comparison of raw extracted entities vs. final merged entities across different agent counts.

As illustrated in Figure 6, the global semantic merging phase compresses the raw extractions ($\sim$119.7K nodes) down to roughly 81.6K to 82.3K unique nodes, achieving a deduplication rate of approximately 31% to 32%. However, the data reveals order-dependent variance introduced by the parallelization: identical inputs of 119.7K extracted nodes at 4 and 8 agents result in slightly different merged node counts (81.7K vs. 82.3K). This discrepancy of roughly 600 nodes confirms that the final graph topology is not strictly invariant to the agent count due to the non-deterministic nature of the shared cache. Furthermore, while overall node counts remain relatively stable, node count equality does not guarantee strict node set equality or identical edge connectivity. A rigorous evaluation of the merging accuracy against a gold-standard benchmark remains a critical area for future work.

5 CONCLUDING REMARKS

Summary. In this paper, we propose *ParaKG*, a parallel LLM-agent framework to scale knowledge graph construction across large scientific corpora. By formulating KG construction as a corpus-level graph integration problem, ParaKG utilizes distributed extraction and hierarchical binary-tree merging to dramatically accelerate end-to-end runtime. Evaluations across open-weight LLMs confirm this architecture achieves high deduplication rates and resolves aliases effectively, though revealing a trade-off between high concurrency and strict structural invariance due to non-deterministic merging.

Limitations and Future Work. Despite significant efficiency gains, our framework has several limitations that present opportunities for future work. First, extraction can be bottlenecked by client-side scheduling caps, preventing full saturation of available LLM agents on shorter documents. Second, the hierarchical reduction tree suffers from a sub-linear scaling bottleneck at the highest levels where intermediate graph sizes grow. Future research can explore distributed scheduling optimizations and novel merging strategies to alleviate these constraints. Finally, while this evaluation focused on computational efficiency, future work could further benchmark knowledge graphs on downstream tasks like domain-specific question answering and hypothesis generation to measure how scalable extraction translates to improved reasoning capabilities.

AI USE STATEMENT

In this work, we used generative AI tools to aid and polish our writing, for retrieval and discovery (e.g., finding related work), and for assistance in code implementation and development. We have not used generative AI tools for generating core research ideas or fabricating data.

We have carefully reviewed all AI-assisted work. Specifically, all AI-generated literature recommendations were manually verified by the authors through independent literature surveys, all AI-assisted text was thoroughly reviewed to ensure technical accuracy and semantic fidelity, and

all AI-generated code was verified and tested for correctness. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Reproducibility Statement

To make our multi-agent framework fully reproducible, we detail the complete pipeline and experimental setup in the main text and supplementary materials. Specifically, Section 3 describes the extraction and merging algorithms, while Appendices A and B provide all agent prompts, system instructions, and inference hyperparameters, such as context limits and θ_{chunk}. Because experiments used open-weight models deployed via vLLM, our throughput metrics in Section 4 can be verified without relying on proprietary APIs. For direct replication, the complete source code will be open-sourced upon acceptance. Due to copyright restrictions, we will provide document DOIs rather than full text for both datasets.

References

Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, Feng Qin, Tianshi Zheng, Xi Peng, Xin Yao, Huiwen Yang, Leijie Wu, Yi Ji, Gong Zhang, Renhai Chen, and Yangqiu Song. AutoSchemaKG: Autonomous Knowledge Graph Construction through Dynamic Schema Induction from Web-Scale Corpora, August 2025. URL https://arxiv.org/abs/2505.23628.

Haonan Bian. LLM-empowered knowledge graph construction: A survey, October 2025. URL https://arxiv.org/abs/2510.20345.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From Local to Global: A Graph RAG Approach to Query-Focused Summarization, April 2024. URL https://arxiv.org/abs/2404.16130v2.

Nadeen Fathallah, Steffen Staab, and Alsayed Algergawy. LLMs4Life: Large Language Models for Ontology Learning in Life Sciences, December 2024. URL https://arxiv.org/abs/2412.02035.

Nadeen Fathallah, Arunav Das, Stefano De Giorgis, Andrea Poltronieri, Peter Haase, and Liubov Kovriguina. NeOn-GPT: A Large Language Model-Powered Pipeline for Ontology Learning. In Albert Meroño Peñuela, Oscar Corcho, Paul Groth, Elena Simperl, Valentina Tamma, Andrea Giovanni Nuzzolese, Maria Poveda-Villalón, Marta Sabou, Valentina Presutti, Irene Celino, Artem Revenko, Joe Raad, Bruno Sartini, and Pasquale Lisena (eds.), *The Semantic Web: ESWC 2024 Satellite Events*, pp. 36–50, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-78952-6. doi: 10.1007/978-3-031-78952-6_4.

Tanjin He, Aikaterini Vriza, Logan Ward, Xu Huang, Yiming Chen, Anubhav Jain, Gerbrand Ceder, Rajeev S. Assary, Ian T. Foster, and Maria K. Y. Chan. Harnessing X-ray Absorption Spectroscopy Data through Multimodal Mining of Battery Literature, August 2026. URL https://arxiv.org/abs/2607.23886.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V. Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-Retriever: Retrieval-Augmented Generation for Textual Graph Understanding and Question Answering, May 2024. URL https://arxiv.org/abs/2402.07630.

Xiaochuan Li, Ryan Ming, Pranav Setlur, Abhijay Paladugu, Andy Tang, Hao Kang, Shuai Shao, Rong Jin, and Chenyan Xiong. Benchmark test-time scaling of general LLM agents, February 2026a. URL https://arxiv.org/abs/2602.18998.

Yaoru Li, Shunyu Liu, Tongya Zheng, Li Sun, and Mingli Song. Parallelized planning-acting for efficient LLM-based multi-agent systems in Minecraft, March 2025. URL https://arxiv.org/abs/2503.03505.

Yuejie Li, Ke Yang, Tao Wang, Bolin Chen, Bowen Li, and Chengjun Mao. Deep GraphRAG: A Balanced Approach to Hierarchical Retrieval and Adaptive Integration, January 2026b. URL https://arxiv.org/abs/2601.11144.

Anna Sofia Lippolis, Miguel Ceriani, Sara Zuppiroli, and Andrea Giovanni Nuzzolese. Ontogenia: Ontology Generation with Metacognitive Prompting in Large Language Models. In Albert Meroño Peñuela, Oscar Corcho, Paul Groth, Elena Simperl, Valentina Tamma, Andrea Giovanni Nuzzolese, Maria Poveda-Villalón, Marta Sabou, Valentina Presutti, Irene Celino, Artem Revenko, Joe Raad, Bruno Sartini, and Pasquale Lisena (eds.), *The Semantic Web: ESWC 2024 Satellite Events*, pp. 259–265, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-78952-6. doi: 10.1007/978-3-031-78952-6_38.

Haitong Luo, Fali Wang, Weiyao Zhang, Xianren Zhang, Zhiwei Zhang, Tianxiang Zhao, Minhua Lin, Jiahao Zhang, Hui Liu, Xianfeng Tang, Qi He, Suhang Wang, Xuying Meng, and Yujun Zhang. Graphs for LLMs: A survey of graph-assisted large language models. *TechRxiv*, 2026. doi: 10.36227/techrxiv.177162088.88045561/v1.

Belinda Mo, Kyssen Yu, Joshua Kazdan, Joan Cabezas, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. KGGen: Extracting knowledge graphs from plain text with language models, November 2025. URL https://arxiv.org/abs/2502.09956.

Mohammad Javad Saeedizade and Eva Blomqvist. Navigating Ontology Development with Large Language Models. In Albert Meroño Peñuela, Anastasia Dimou, Raphaël Troncy, Olaf Hartig, Maribel Acosta, Mehwish Alam, Heiko Paulheim, and Pasquale Lisena (eds.), *The Semantic Web*, pp. 143–161, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-60626-7. doi: 10.1007/978-3-031-60626-7_8.

Jiaqi Sun, Shiyou Qian, Zhangchi Han, Wei Li, Zelin Qian, Dingyu Yang, Jian Cao, and Guangtao Xue. LKD-KGC: Domain-Specific KG Construction via LLM-driven Knowledge Dependency Parsing, May 2025a. URL https://arxiv.org/abs/2505.24163.

Yushi Sun, Kai Sun, Yifan Ethan Xu, Xiao Yang, Xin Luna Dong, Nan Tang, and Lei Chen. KERAG: Knowledge-Enhanced Retrieval-Augmented Generation for Advanced Question Answering, September 2025b. URL https://arxiv.org/abs/2509.04716.

Yash Tiwari, Owais Ahmad Lone, and Mayukha Pal. OntoRAG: Enhancing Question-Answering through Automated Ontology Derivation from Unstructured Knowledge Bases, May 2025. URL https://arxiv.org/abs/2506.00664.

Qingyun Wang, Doug Downey, Heng Ji, and Tom Hope. SciMON: Scientific Inspiration Machines Optimized for Novelty. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 279–299, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.18.

Chengxuan Xia, Qianye Wu, Sixuan Tian, and Yilun Hao. Parallelism meets adaptiveness: Scalable documents understanding in multi-agent LLM systems, July 2025. URL https://arxiv.org/abs/2507.17061.

Hongbin Ye, Honghao Gui, Xin Xu, Xi Chen, Huajun Chen, and Ningyu Zhang. Schema-adaptable Knowledge Graph Construction, November 2023. URL https://arxiv.org/abs/2305.08703.

Bowen Zhang and Harold Soh. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9820–9836, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.548.

Enhao Zhang, Erkang Zhu, Gagan Bansal, Adam Fourney, Hussein Mozannar, and Jack Gerrits. Optimizing sequential multi-step tasks with parallel LLM agents, July 2025. URL https://arxiv.org/abs/2507.08944.

Xinjie Zhao, Moritz Blum, Fan Gao, Yingjian Chen, Boming Yang, Luis Marquez-Carpintero, Monica Pina-Navarro, Yanran Fu, So Morikawa, Yusuke Iwasawa, Yutaka Matsuo, Chanjun Park, and Irene Li. AGENTiGraph: A multi-agent knowledge graph framework for interactive, domain-specific LLM chatbots. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6757–6761. Association for Computing Machinery, 2025. doi: 10.1145/3746252.3761459.

Zulun Zhu, Tiancheng Huang, Kai Wang, Junda Ye, Xinghe Chen, and Siqiang Luo. Graph-Based Approaches and Functionalities in Retrieval-Augmented Generation: A Comprehensive Survey. *ACM Computing Surveys*, 58(10):261:1–261:38, April 2026. ISSN 0360-0300. doi: 10.1145/3795880.

A PROMPTS FOR KG CONSTRUCTION

This section provides the exact system prompts used to extract knowledge graphs from the raw text and subsequently merge them.

The semantic behavior and accuracy of the LLM across this pipeline are entirely governed by a custom, comprehensive zero-shot domain context prompt that we developed. This domain context is injected into the extraction engine to guide the LLM's understanding of what constitutes a valid scientific entity or relation. As shown below, the prompt heavily emphasizes materials science and battery-specific terminology, ensuring that the model prioritizes central scientific contributions, characterization methods, and performance metrics over generic bibliographic metadata.

Domain Context for KG Extraction

```
Scientific paper in materials science, with emphasis on
battery materials, energy-storage materials, computational
materials discovery, and materials informatics.  Build a clean,
contribution-focused knowledge graph for the main paper content
only.  In addition to extracting the paper's own contributions,
explicitly perform scientific information extraction over
the title, abstract, introduction, and background/motivation
sentences.  Prioritize entities that are central to the
paper's materials-science contribution:  target materials,
chemical formulas, compositions, dopants, defects, phases,
crystal structures, electrodes, electrolytes, synthesis routes,
characterization methods, machine-learning models, target
properties, and battery performance metrics.  Extract relations
that express materials semantics, such as has composition,
contains element, doped with, has phase, has crystal structure,
synthesized by, forms intermediate, transforms into, reacts with,
intercalates, transports ion, conducts, catalyzes, degrades by,
stabilizes, predicts, trained on, evaluated on, measured by,
achieves capacity, has voltage, has conductivity, used-for, and
compared against.  Favor precise relations connecting materials,
methods, structure-processing-property-performance relationships,
datasets, models, experiments, and reported findings.
```

The initial KG extraction utilizes a standard two-pass algorithm (Mo et al., 2025) to maximize structural consistency. In the first pass, the LLM is prompted to comprehensively identify all relevant entities within a given text chunk. In the second pass, the LLM receives the text alongside the previously extracted entities and is prompted to formulate the relations connecting them.

Two-Pass Extraction Mechanics

Pass 1 (Entity Extraction):

```
Extract key entities from the source text.  Extracted entities are
subjects or objects.  This is for an extraction task, please be
THOROUGH and accurate to the reference text.
```

Pass 2 (Relation Extraction):

```
Extract subject-predicate-object triples from the source text.
Subject and object must be from entities list.  Entities provided
were previously extracted from the same source text.  This is for
an extraction task, please be thorough, accurate, and faithful to
the reference text.
```

After extracting the entities and relations from each chunk of text, the local graphs are aggregated and we begin the global deduplication process. This semantic merging phase is performed by evaluating entity and relation clusters using the following system prompt.

System Prompt for Semantic Entity Resolution

```
Merge scientific-paper knowledge graphs in materials science,
with emphasis on battery materials, energy-storage materials,
computational materials discovery, and materials informatics.
Canonicalize only clearly equivalent aliases, acronym/full-name
variants, chemical-formula formatting variants, spelling
variants, Unicode/subscript/superscript variants, and minor
OCR or hyphenation artifacts.  Merge common battery/materials
abbreviations with their full names when equivalent, such as
lithium-ion battery/LIB, density functional theory/DFT, graph
neural network/GNN, and solid electrolyte interphase/SEI.
Preserve distinctions between related-but-different materials,
compositions, phases, dopants, defects, electrodes, electrolytes,
interfaces, datasets, models, synthesis routes, characterization
methods, and mechanisms.  Do not merge a broad class with a
specific material or property, for example cathode material vs
LiFePO4, electrolyte vs solid electrolyte, capacity vs capacity
retention, or DFT calculation vs DFT dataset.
```

B Example Text Chunks from Document Segmentation

To process full-length scientific literature through the LLM, each document is systematically divided into manageable segments. In our pipeline, the extracted text is partitioned with a strict maximum chunk size of 4000 characters, with boundaries constrained to sentence endings to preserve local grammatical coherence. This character threshold ensures that the LLM has sufficient local context to resolve complex cross-sentence relationships while remaining well within its optimal context window to prevent attention degradation.

Because we prioritize high-throughput text extraction over computationally expensive, vision-based OCR engines, the resulting text inherently retains PDF parsing artifacts—such as inline citations merging into chemical formulas (e.g., TiO2.14) or publisher metadata interrupting paragraphs. Rather than relying on rigid upstream normalization to fix these issues, our framework shifts the burden of structural recovery and regularization to the LLM extraction agents. Below is an illustrative snippet of a raw, 4000-character text chunk generated from a solid-state Li-S battery paper in the Battery-173 corpus. It demonstrates both the dense concentration of domain terminology targeted by our extraction prompts and the raw formatting noise that our LLM pipeline successfully regularizes during graph construction:

Example Chunk Snippet

```
However, the issue of polysulfide shuttling with its corresponding
capacity fading is a major impediment to its commercialization.
Replacing traditional liquid electrolytes with solid-state
electrolytes (SEs) is a potential solution.  Here, we present
a comprehensive study of the thermodynamics and kinetics of
the cathode-electrolyte interface in all-solid-state LSBs using
density functional theory based calculations and a machine
learning interatomic potential.  We find that among the major
solid electrolyte chemistries (oxides, sulfides, nitrides, and
halides), sulfide SEs are generally predicted to be the most
stable against the S8 cathode, while the other SE chemistries
are predicted to be highly electrochemically unstable.  If the
use of other SE chemistries is desired for other reasons, several
binary and ternary sulfides (e.g., LiAlS2, Sc2S3, Y2S3) are
predicted to be excellent buffer layers.  Finally, an accurate
moment tensor potential to study the S8|β-Li3PS4 interface was
developed using an active learning approach.  Molecular dynamics
(MD) simulations of large interface models (>1000s atoms) revealed
that the most stable Li3PS4(100) surface tends to form interfaces
```

```
with S8 with 2D channels and lower activation barriers for Li
diffusion.  These results provide critical new insights into
the cathode-electrolyte interface design for next-generation
all-solid-state LSBs.  INTRODUCTION Lithium-sulfur batteries
(LSBs) have garnered immense interest as a potential alternative to
lithium-ion batteries (LIBs) for future energy storage needs due to
their extremely high theoretical capacity (1672 mAh g-1)1,2 and low
cost due to the use of abundant sulfur.3 However, commercialization
of LSBs still requires solving several challenges that are
intrinsic to this technology.  While other challenges such as
the electrically insulating nature and large volume change during
cycling (∼80%) of the S cathode are being addressed via structure
and composition engineering,4-6 the polysulfide shuttling problem,
which is considered to be the most detrimental factor to the cell
performance, has not been completely mitigated.  Polysulfide
shuttling refers to the concentration gradient-driven shuttling
of soluble higher order polysulfides and insoluble lower order
polysulfides between the cathode and anode through the liquid
electrolyte medium, resulting in irreversible loss of the cathode
and anode material.  While the formation of polysulfides can be
mitigated to some extent with alternative cathode chemistries,
these solutions typically result in loss of energy density.7-9
Replacing the liquid electrolyte with a solid electrolyte (SE) has
been proposed as a potential solution to address the polysulfide
shutting problem in LSBs.10 For example, gel polymer electrolytes
use polar molecules such as polyvinylidene fluoride11 (PVDF) and
poly(methyl methacrylate)12 (PMMA) to trap dissolved polysulfides.
In solid polymer electrolytes, the polysulfide migration is
inhibited by their adsorption onto inorganic filler surfaces
such as Al2O3 13 and TiO2.14 Finally, composite SEs made by
dispersing inorganic SEs in poly(ethylene oxide) (PEO)-based
electrolytes15-17 have been found to have a better ability to
inhibit polysulfide movement and improve interfacial stability and
mechanical properties when compared to single all-solid polymer or
inorganic SEs.10 In addition, an all-solid-state LSB architecture
may yield other potential advantages of improved system-level
energy density and safety.18,19 In contrast to all-solid-state
LIBs,20-27 the interfaces in all-solid-state LSBs, especially
that between the SE and the S cathode, have not been extensively
studied.
```

By segmenting the corpus into these section-aware, ∼4000-character blocks, our pipeline effectively balances global scalability with local context preservation. This distributed approach enables thousands of extraction tasks to run in parallel without sacrificing the precision required for complex entity resolution, such as mapping measured properties to synthesis conditions.