COVERKG: COVERAGE AND COMPACTNESS DUAL REWARDS FOR KNOWLEDGE GRAPH EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge-graph-based retrieval-augmented generation relies on an LLM extractor to build a graph index, the quality of which directly governs retrieval effectiveness and storage efficiency. Yet existing methods supervise the extractor only through downstream QA accuracy. This sparse, end-to-end signal cannot diagnose whether failures arise from insufficient semantic coverage or from redundant extractions that introduce noise into retrieval. We introduce CoverKG, a framework that characterizes index quality through two complementary objectives, namely coverage and compactness, and converts this characterization into a construction-time training signal that can be computed for every source chunk and used to optimize the extractor with reinforcement learning. Coverage aligns the source sentences with the extracted graph elements to pinpoint missing semantics at the sentence level. Compactness evaluates each element's marginal contribution relative to the other elements in the same extraction, directly penalizing redundancy. A composite reward balancing both objectives drives policy optimization. The framework is readily applicable to the graph construction stage of various graph-based RAG systems. Experiments across multiple graph-based RAG pipelines and QA benchmarks show that, even under the same downstream evaluation protocol as prior work, extractors trained with CoverKG consistently achieve higher accuracy with more compact graph indices, demonstrating that per-chunk, dual-objective supervision at index construction time yields both more effective retrieval and more efficient storage.

1 INTRODUCTION

Retrieval-augmented generation (RAG) has emerged as a standard paradigm for grounding large language models (LLMs) in external corpora, enhancing factual accuracy and incorporating domain knowledge without updating model parameters (Lewis et al., 2020). A prominent line of research moves beyond flat chunk retrieval by organizing a corpus into structured entity–relation graphs, enabling multi-hop evidence discovery, entity-centric retrieval, and global sensemaking in knowledge-graph-based RAG pipelines (Edge et al., 2024; Guo et al., 2025; Gutiérrez et al., 2025). In this setting, the constructed graph serves as a queryable index whose quality directly governs retrieval effectiveness and storage efficiency: it determines which source evidence remains addressable, how much irrelevant structure downstream operators must process, and ultimately the reliability and cost of retrieval.

Yet graph construction remains an unrecoverable bottleneck: semantics omitted during extraction cannot be recovered downstream. Existing chunk-wise LLM extraction suffers from insufficient semantic coverage and redundant elements (Figure 1), which respectively make evidence unretrievable and introduce retrieval noise. Recent RL methods instead use downstream QA accuracy to supervise agents or graph builders (Tsang et al., 2026; Luo et al., 2025; Yu et al., 2026; Li et al., 2025). Because this sparse reward cannot localize either deficiency, the extractor requires a construction-time supervision signal available for every source chunk.

Under the retrieval semantics of RAG, one reliable graph element suffices to index a source semantic unit. A high-quality index should therefore maximize semantic coverage while minimizing redundant elements. We ask whether these objectives—coverage and compactness—can provide a construction-time reward for every source chunk to train the extractor via reinforcement learning.

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.

CoverKG answers this question by evaluating each candidate extraction directly against its source chunk. Coverage matches every source sentence to its strongest thresholded graph element, following the single-hit sufficiency principle that one aligned element suffices to make a sentence retrievable, while compactness penalizes elements with little marginal contribution to coverage. Their F_β-style combination yields a bounded reward that discourages both missing semantics and redundant extractions, and drives extractor optimization through GRPO (Shao et al., 2024). Unlike end-task rewards, its components provide local diagnostics for uncovered sentences and low-utility elements, and the resulting sequence-level reward can be integrated into various graph-based RAG systems.

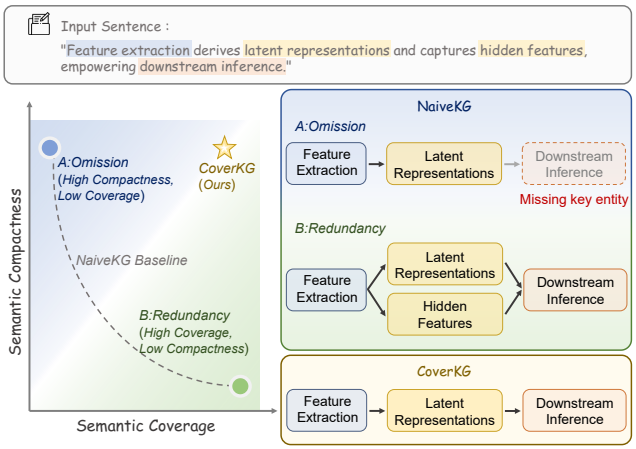

Figure 1: Comparison of knowledge graph extraction paradigms. NaiveKG methods suffer from either omission (A) or redundancy (B), whereas CoverKG jointly attains high semantic coverage and compactness.

In summary, this paper makes the following contributions:

- We formulate LLM-based knowledge graph extraction for RAG as a *coverage–compactness* problem and introduce a per-chunk, construction-time supervision signal that replaces sparse downstream QA accuracy; its diagnostic components explicitly localize uncovered source sentences and low-utility graph elements.
- We propose a composite reward that reconciles coverage maximization with compactness enforcement within a single tunable F_β formulation, integrating thresholded coverage matching, leave-one-out marginal utility, and participation ratio regularization into one unit-bounded signal. This formulation yields a sequence-level training signal for every source chunk that is compatible with GRPO-based policy optimization.
- We deploy the trained extractor as a drop-in component for mainstream graph-based RAG frameworks, achieving gains on 12 of 15 benchmark configurations (up to $+8.97$ F1) while reducing index nodes by 48% and edges by 70% on average across the three RAG backends, confirming that dual-objective supervision simultaneously improves retrieval effectiveness and storage efficiency.

2 Related Work

LLM-based graph construction commonly controls extraction quality through engineering heuristics. Prompting uses structured formats and demonstrations (Zhu et al., 2024), while incremental extraction integrates entities and relations across documents (Lairgi et al., 2025). Iterative extraction recovers omitted entities through multiple passes (Edge et al., 2024). Post-hoc canonicalization merges redundant nodes via string matching or embedding-based clustering (Vashishth et al., 2018). These strategies improve graph quality but do not directly train extraction policies against an index-quality objective.

We therefore organize related efforts that do attempt principled supervision by where the quality signal originates: against an external reference standard (§2.1), at retrieval or reasoning time (§2.2), or through downstream QA accuracy fed back to the constructor (§2.3).

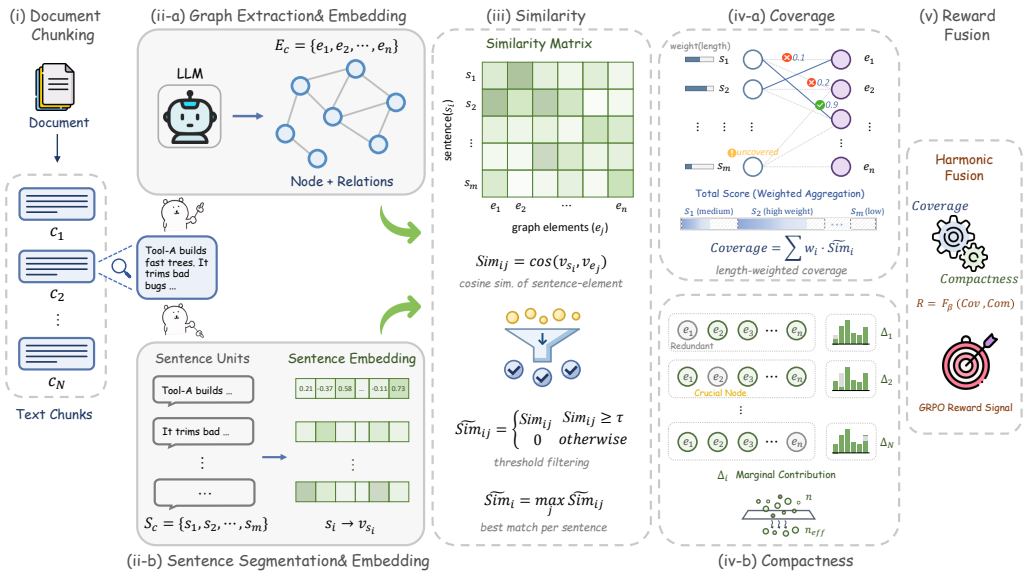

Figure 2: The CoverKG reward pipeline. A document is (i) chunked and encoded along two paths—(ii-a) LLM-based graph extraction and (ii-b) sentence embedding. From the (iii) thresholded similarity matrix, we compute (iv-a) Coverage (length-weighted sentence coverage) and (iv-b) Compactness (redundancy penalty via marginal contribution), then (v) fuse them harmonically (F_β) into the GRPO reward R.

2.1 Supervising Against a Reference Standard

Reference-based quality frameworks measure completeness against required schema and population information (Zaveri et al., 2016). Canonicalization methods such as CESI cluster coreferent entity and relation phrases and evaluate against gold canonical entities (Vashishth et al., 2018). Such references are generally unavailable at chunk level, where extraction quality instead depends on whether graph elements make source semantics retrievable.

2.2 Optimizing Retrieval and Reasoning

Self-RAG improves retrieval and generation through learned reflection on when to retrieve and whether evidence supports an answer (Asai et al., 2024). Graph-R1 and GraphRAG-R1 optimize query-time retrieval and reasoning through reinforcement learning while treating the constructed graph as given (Luo et al., 2025; Yu et al., 2026). These methods improve the use of indexed evidence but do not directly supervise the extractor.

2.3 Supervising Graph Construction Through Downstream QA

AutoGraph-R1 optimizes graph construction using downstream RAG utility (Tsang et al., 2026). Agentic-KGR couples LLM and graph updates through multi-round RL, dynamic schema expansion, and retrieval-augmented memory (Li et al., 2025). DeepRefine instead learns query-conditioned refinement of an existing knowledge base using a Gain-Beyond-Draft reward (Huang et al., 2026). These approaches obtain supervision from downstream task outcomes rather than direct evaluation of individual chunk extractions. Such signals do not directly distinguish insufficient semantic coverage from redundant graph elements. Their training objectives are also coupled to the downstream evaluation pipeline.

Our position. The fundamental gap across all three lines of work is the absence of a construction-time training signal that directly characterizes index quality at the granularity of individual source chunks. CoverKG fills this gap by optimizing the extraction policy against two complementary objectives—coverage and compactness—before the graph is ever queried.

3 PRELIMINARIES

We study the graph construction stage of a knowledge-graph-based RAG pipeline in isolation, so that index quality can be characterized and optimized at the source, before any downstream module consumes the graph.

Graph construction as extraction. A corpus is partitioned into document chunks, and each chunk c is the unit at which the graph is built. We segment c into *semantic units* $U_c = \{s_1, \ldots, s_m\}$, realized as sentences—the atomic pieces of source meaning that must remain recoverable through the index. An LLM-based extractor parameterized by θ reads c and emits a set of *graph elements* $E_c = \{e_1, \ldots, e_n\}$, where each e_j is a textual description of an entity or a relation. The union $\bigcup_c E_c$ forms the knowledge graph index over the corpus.

Semantic matching. A shared text encoder maps any unit or element to an ℓ_2-normalized embedding, written $\mathbf{v}_{s_i}$ and $\mathbf{v}_{e_j}$. Since retrieval locates source units via the graph elements that index them, we regard an alignment as a *retrievable link* only when its cosine similarity $\langle \mathbf{v}_{s_i}, \mathbf{v}_{e_j} \rangle$ exceeds a threshold τ, mirroring the nearest-neighbor cutoffs used at query time.

Problem statement. No gold graph exists at chunk granularity, so we seek a reference-free scalar reward $R(c, E_c) \in [0, 1]$ computed only from c, E_c, and their embeddings. We cast construction as reinforcement learning: the extractor is modeled as a policy π_θ and optimized against R. The reward should be high when E_c faithfully covers the semantics of c with a compact, nonredundant set of elements.

4 METHODOLOGY

4.1 OVERVIEW

CoverKG characterizes index quality using two complementary objectives that trade off against each other—coverage and compactness—and converts this characterization into a construction-time training signal for every source chunk to optimize the extractor via reinforcement learning (Figure 2). We require its components to provide local diagnostics, identifying *which source sentence lacks coverage* and *which element has low marginal utility*. A coverage term Γ rewards indexing every source sentence with at least one element. A compactness term C penalizes elements whose marginal contribution relative to the other elements in the same extraction is negligible. A composite objective combines them into a single reward that drives policy optimization.

4.2 COVERAGE–COMPACTNESS REWARD

Semantic coverage matrix. Using the ℓ_2-normalized embeddings, the alignment between source sentence s_i and element e_j is the cosine similarity $\mathrm{Sim}_{ij} = \langle \mathbf{v}_{s_i}, \mathbf{v}_{e_j} \rangle \in [-1, 1]$. A weak match should not count as a retrieval-capable link, so CoverKG keeps only alignments above the threshold τ:

$$\widetilde{\mathrm{Sim}}_{ij} = \begin{cases} \mathrm{Sim}_{ij}, & \mathrm{Sim}_{ij} \geq \tau, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Thresholding mirrors the nearest-neighbor cutoffs applied at query time; since $\tau \geq 0$, the truncation guarantees $\widetilde{\mathrm{Sim}}_{ij} \in [0, 1]$, the non-negativity that the coverage function relies on. It is one of only two reward hyperparameters; its selection—determined via Otsu's method (Otsu, 1979) on a high-quality reference dataset—and the encoder choice are described in §5.1.

Coverage. Coverage aligns source sentences with extracted graph elements to pinpoint missing semantics at the sentence level. A source sentence is recoverable as soon as *one* element indexes it, so CoverKG scores its coverage ϕ by the strongest surviving alignment,

$$\phi(s_i; E_c) = \max_{1 \leq j \leq n} \widetilde{\mathrm{Sim}}_{ij}. \tag{2}$$

This *single-hit sufficiency* matches the recall-first semantics of RAG and makes coverage insensitive to duplicate elements, delegating redundancy control to the compactness term. Chunk-level

coverage Γ is the length-weighted average of per-sentence coverage:

$$w_i = \frac{\sqrt{\text{len}(s_i)}}{\sum_{k=1}^{m} \sqrt{\text{len}(s_k)}}, \tag{3}$$

$$\Gamma(E_c) = \sum_{i=1}^{m} w_i \, \phi(s_i; E_c), \tag{4}$$

where the concave square-root weighting gives more informative sentences greater influence while damping the tendency of a few long sentences to dominate. Since $\widetilde{\text{Sim}}_{ij} \in [0, 1]$ and the weights form a convex combination, $\Gamma(E_c) \in [0, 1]$. Moreover, Γ is monotone and submodular in E_c, which corresponds to a weighted facility-location form whose marginal gains admit memoized evaluation (Lin & Bilmes, 2011; Iyer & Bilmes, 2019), so removing an element never increases coverage.

Marginal contribution compactness. Compactness penalizes redundancy through each element's leave-one-out marginal $\Delta_j = \Gamma(E_c) - \Gamma(E_c \setminus \{e_j\}) \geq 0$ (Conforti & Cornuéjols, 1984). Non-negativity follows from the monotonicity of Γ: an element is useful to the extent that its removal lowers sentence coverage, whereas redundant elements have $\Delta_j \approx 0$. All marginals can be computed exactly in $O(mn)$ time using each sentence's top-two thresholded alignments (Appendix A.1).

We aggregate the marginals into an *effective number of useful elements* via a participation ratio, and define compactness C as its fraction of the extraction:

$$n_{\text{eff}} = \frac{\left(\sum_j \Delta_j\right)^2}{\sum_j \Delta_j^2}, \tag{5}$$

$$C(E_c) = \frac{n_{\text{eff}}}{n} \in [0, 1]. \tag{6}$$

We use the convention $n_{\text{eff}} = 0$ when all marginals vanish, and set $C(E_c) = R(c, E_c) = 0$ when $n = 0$. The participation ratio reaches its maximum $n_{\text{eff}} = n$ when all marginals are uniform and approaches 1 as contributions concentrate on a single element, thus providing a differentiable measure of how many elements contribute meaningfully without relying on a brittle hard count. Duplicating an element adds a near-zero Δ, leaving n_{eff} almost unchanged while increasing n, so compactness falls. The two objectives use complementary normalizations: Γ averages over the m source sentences, while C divides by the n extracted elements. Consequently, increasing the number of extracted elements cannot raise the reward without adding new coverage, and a redundant element is penalized solely through the compactness channel, not double counted in both terms.

Reward. The composite reward balancing both objectives drives policy optimization. CoverKG combines the two terms in a weighted harmonic mean analogous to the F_β score:

$$R(c, E_c) = \left(1 + \beta^2\right) \frac{\Gamma(E_c)\, C(E_c)}{\beta^2\, C(E_c) + \Gamma(E_c) + \epsilon}. \tag{7}$$

Here, $\epsilon > 0$ is a small numerical stabilizer that prevents a $0/0$ indeterminate when both terms vanish. The parameter β controls the relative emphasis: $\beta > 1$ upweights coverage, whereas $\beta < 1$ favors compactness; sensitivity to β is examined in §5.4. The harmonic form encourages *both* terms high, so padding the output raises no term and lowers C. The reward components also provide local diagnostics: a low $\phi(s_i; E_c)$ localizes a sentence with insufficient coverage and a low Δ_j identifies a redundant element. Encoding costs $O((m+n)d)$ and the alignment matrix $O(mnd)$ for embedding dimension d, after which coverage, all marginals (Appendix A.1), and the reward are computed in $O(mn)$—the same order as a single matrix multiplication, adding negligible overhead to the extraction.

4.3 Policy Optimization with GRPO

Following supervised initialization (Ouyang et al., 2022), we optimize the SFT checkpoint with Group Relative Policy Optimization (GRPO) (Shao et al., 2024). For each chunk c, we sample G candidate extractions from $\pi_{\theta_{\text{old}}}$, score them with Eq. (7), and standardize rewards within the group to obtain advantages. All tokens of a candidate share its sequence-level advantage. We optimize the clipped GRPO objective with KL regularization toward the SFT reference policy. The full objective and training settings are provided in Appendices A.2 and B.1, respectively.

5 EXPERIMENTS

Our evaluation tests the central claim of CoverKG: that per-chunk, dual-objective supervision at index construction time yields both more effective retrieval and more efficient storage, and that this improvement transfers across diverse graph-based RAG pipelines independently of the backend. We organize the study around three research questions.

- **RQ1 (Effectiveness).** Does an extractor trained with CoverKG improve downstream QA accuracy across heterogeneous backends and across both single-hop and multi-hop tasks?
- **RQ2 (Reward design).** How do the coverage and compactness objectives and the balance parameter β each contribute to index quality and downstream utility?
- **RQ3 (Efficiency).** Does the more compact graph index translate into more efficient storage at deployment time without sacrificing downstream quality?

5.1 EXPERIMENTAL SETUP

Baselines. We compare along two axes. The *training objective* axis holds the extractor architecture fixed and isolates the effect of the CoverKG reward: **Base** denotes the untrained instruction-tuned extractor, **SFT** adds supervised fine-tuning on the extraction corpus, and **CoverKG** further optimizes the SFT checkpoint with our composite reward under GRPO. To decouple the reward from any single retrieval architecture, every trained extractor is plugged into three structurally distinct *backends*—**GraphRAG** (Edge et al., 2024), **LightRAG** (Guo et al., 2025), and **HippoRAG2** (Gutiérrez et al., 2025)—which differ in graph schema, retrieval mechanism, and query-time traversal strategy.

Datasets and Evaluation Metrics. *Training data.* CoverKG requires no gold graphs. The extractor is first trained via supervised fine-tuning on a Wikipedia-derived extraction set (3K samples) to initialize the extractor with a valid output format, then optimized with GRPO on source corpora from HotpotQA (15K samples) and MuSiQue (1K samples) (Yang et al., 2018; Trivedi et al., 2022); the reward is computed only from each chunk and its extraction (§4.2), so the QA annotations are never used as supervision. *Evaluation benchmarks.* We report on five benchmarks in two regimes: *single-hop / factual* QA on Natural Questions and PopQA (Kwiatkowski et al., 2019; Mallen et al., 2023), and *multi-hop reasoning* QA on HotpotQA, 2WikiMultiHopQA (2Wiki), and MuSiQue (Yang et al., 2018; Ho et al., 2020; Trivedi et al., 2022). For domain-level analysis, we additionally adopt the Medical domain of GraphRAG-Bench (Xiang et al., 2026), which spans fact retrieval, complex reasoning, contextual summarization, and creative generation, evaluating both graph construction and final generation quality. *Metrics.* We report token-level F1 (Rajpurkar et al., 2016) for answer quality on the five QA benchmarks. On GraphRAG-Bench, we follow the benchmark protocol and report answer correctness (ACC), ROUGE-L, coverage (Cov.), and faithfulness (Faith.) per question type. At the graph level, we report node and edge counts, average degree, and clustering coefficient to quantify structural efficiency.

Implementation Details. We use Qwen3-8B (Yang et al., 2025) as the extractor and Qwen3-Embedding-4B (Zhang et al., 2025) as the encoder. The retrieval threshold is τ=0.63, selected by Otsu's method on a high-quality reference dataset. We default to β=1 and sweep $\{0.5, 1, 2\}$ in §5.4. Training hyperparameters and the compute environment are reported in Appendix B.1.

5.2 MAIN RESULTS

Per-backend analysis. Under *GraphRAG*, CoverKG achieves the best F1 on all five benchmarks; the largest gain is on PopQA (+2.67 over Base, +1.86 over SFT). Under *LightRAG*, CoverKG achieves the best performance on three of five benchmarks, with HotpotQA and MuSiQue being the exceptions; nonetheless, it still substantially mitigates SFT's severe degradation on HotpotQA—raising F1 from 6.11 to 11.88. Within the *HippoRAG2* framework, CoverKG yields improvements on four out of five benchmarks, with only a marginal difference of 0.48 points on NQ.

Table 1 and Figure 3 address RQ1 from two complementary angles: downstream QA quality and graph-index size. CoverKG improves over Base on 12 of 15 backend–benchmark cells, with positive gains ranging from +0.18 F1 (GraphRAG/2Wiki) to +8.97 F1 (LightRAG/PopQA). These gains appear across three structurally distinct backends, indicating that they stem from index quality rather than from a particular retrieval mechanism. A breakdown by task regime is provided in Appendix C.

Table 1: QA results (token-level F1). Best result per backend and benchmark is bolded.

Method	Extractor	Single-hop		Multi-hop		
		NQ	PopQA	HotpotQA	2Wiki	MuSiQue
GraphRAG	Base	12.93	17.38	13.58	12.10	8.18
	SFT	13.65	18.19	12.35	10.86	7.58
	CoverKG	**13.73**	**20.05**	**14.72**	**12.28**	**8.80**
LightRAG	Base	12.99	9.72	**12.81**	11.27	**7.23**
	SFT	12.90	12.08	6.11	7.63	3.36
	CoverKG	**13.77**	**18.69**	11.88	**11.53**	6.85
HippoRAG2	Base	**43.14**	69.66	69.06	55.32	35.80
	SFT	42.71	70.24	71.44	57.64	37.14
	CoverKG	42.66	**70.61**	**72.01**	**58.56**	**37.17**

SFT failure mode. SFT degrades LightRAG by up to 6.70 F1 on HotpotQA and 3.64 F1 on 2Wiki, indicating that format-only supervision can produce graphs misaligned with a backend's retrieval heuristic. CoverKG's quality signal provides calibration beyond format compliance: it substantially recovers the SFT degradation on HotpotQA and surpasses Base on 2Wiki.

Graph size. Figure 3 shows that CoverKG produces consistently smaller graphs across all corpora and backends, despite the F1 improvements above. This confirms that the compactness objective prunes redundant—rather than informative—elements: the extractor learns to emit fewer, higher-utility graph elements during construction.

Regressions. The three cases where Base outperforms CoverKG are LightRAG/HotpotQA (−0.93), LightRAG/MuSiQue (−0.38), and HippoRAG2/NQ (−0.48). These regressions remain limited in absolute terms and occur in settings where the base extractor's more verbose output already aligns well with the backend's retrieval heuristic.

5.3 Domain-Level Evaluation

To validate generalization, we evaluate on the Medical domain of GraphRAG-Bench (Xiang et al., 2026), an independent benchmark that assesses both graph construction quality and generation quality across four question types. Table 2 reports the resulting index topology; Table 3 reports generation quality.

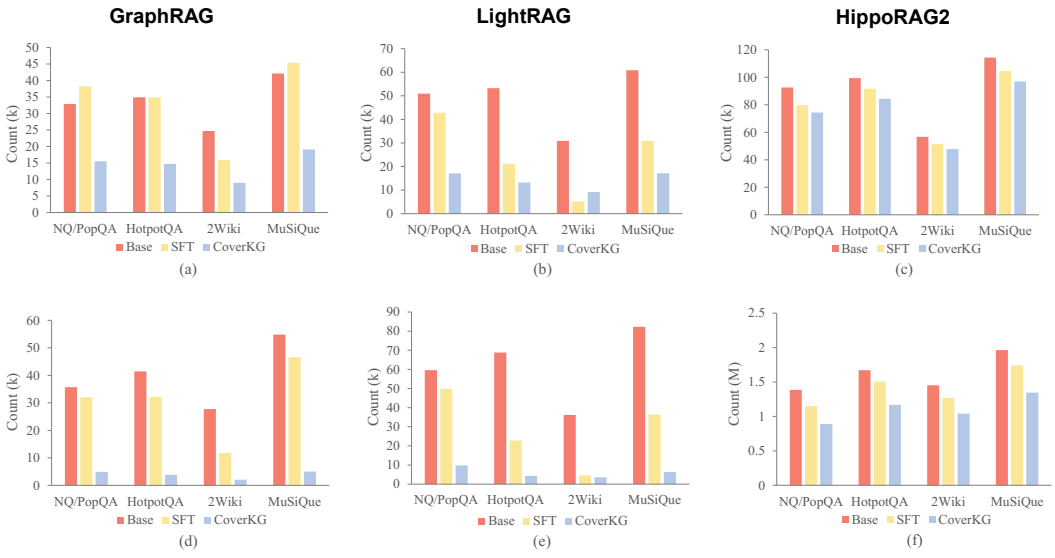

Figure 3: Node and edge counts across backends and corpora for Base, SFT, and CoverKG.

Index topology. Table 2 quantifies the structural compression on GraphRAG-Bench. Under GraphRAG, CoverKG reduces node count by 70% (3277→999) and edge count by 97% (3168→81), yielding a far sparser index. Under HippoRAG2, reductions are 40% for nodes (8132→4921) and 65% for edges (86956→30082).

Table 2: Index topology on the GraphRAG-Bench Medical domain corpora.

Metric	Base	CoverKG
GraphRAG		
Nodes	3277	999
Edges	3168	81
Average Degree	1.930	0.162
Avg. Clust. Coeff	0.340	0.067
HippoRAG2		
Nodes	8132	4921
Edges	86956	30082
Average Degree	21.14	12.23
Avg. Clust. Coeff	0.474	0.547

Generation quality. Table 3 shows that CoverKG improves generation quality on the majority of metrics despite the dramatic graph compression. Under GraphRAG, CoverKG leads on 7 of 9 metrics; the largest gains appear on structured tasks such as Contextual Summarization Cov. (+2.29) and Complex Reasoning ROUGE-L (+1.37). Under HippoRAG2, CoverKG leads on 8 of 9 metrics, with the largest gain on Faithfulness (+2.02). The regressions concentrate on Creative Generation ACC (-0.17 for GraphRAG, -1.47 for HippoRAG2) and GraphRAG Creative Generation Cov. (-1.70): creative open-ended tasks benefit from breadth of context rather than the focused, factual coverage that CoverKG optimizes. These results reinforce the main-experiment finding: CoverKG achieves quality improvement and graph compression simultaneously.

5.4 Ablation Study

Table 4 addresses RQ2: we train the extractor with one reward component removed and evaluate the resulting graph on GraphRAG. Removing the coverage term causes the larger degradation: an average drop of 1.05 F1 across benchmarks (largest on PopQA: -1.73). Without the coverage signal, the extractor produces graphs with insufficient semantic coverage, leaving key source sentences with no matching element in the index, so the evidence is never retrievable regardless of the backend. Removing the compactness term causes a smaller but consistent drop of 0.43 F1 on average (largest on PopQA: -0.93); without compactness, the extractor emits redundant elements that inflate the graph and introduce retrieval noise, diluting the relevant evidence among near duplicates. The full dual-objective reward outperforms both ablations on every benchmark: the two objectives address complementary failure modes (omission vs. redundancy), and neither subsumes the other.

Effect of β. Figure 4 traces the coverage–compactness tradeoff as β varies over $\{0.5, 1, 2\}$. A value of $\beta<1$ over-penalizes subgraph size at the expense of coverage, producing sparse structures that omit bridging entities essential for multi-hop reasoning and consequently degrading F1. Setting $\beta>1$ overweights coverage and permits some redundancy without corresponding downstream gain. The balanced setting $\beta=1$ achieves the best overall performance, indicating that equal emphasis on both objectives is optimal for the RAG extraction setting.

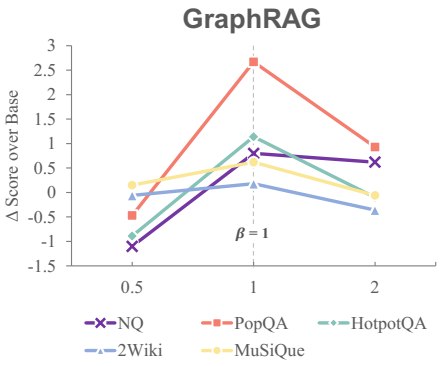

Figure 4: Downstream QA performance as the balance parameter β varies.

5.5 Efficiency Analysis

RQ3 asks whether the construction-time reward translates into more efficient storage at deployment. Our experiments confirm this. On the main QA benchmarks, Figure 3 shows that CoverKG consistently reduces node and edge counts across all corpora and backends. Specifically, the average node reduction reaches 57%, 71%, and 16% for GraphRAG, LightRAG, and HippoRAG2, respectively, while the corresponding edge reduction amounts to 90%, 89%, and 31%.

Table 3: Medical-domain generation quality on GraphRAG-Bench. ACC: answer correctness; Cov.: coverage; Faith.: faithfulness. Best result per backend and metric is bolded.

Extractor	Fact Retrieval		Complex Reasoning		Contextual Summ.		Creative Gen.		
	ACC	ROUGE-L	ACC	ROUGE-L	ACC	Cov.	ACC	Cov.	Faith.
GraphRAG									
Base	65.98	31.76	71.05	26.39	72.22	49.69	**63.06**	**39.82**	40.32
CoverKG	**66.23**	**32.97**	**71.56**	**27.76**	**73.50**	**51.98**	62.89	38.12	**41.57**
HippoRAG2									
Base	70.52	37.20	73.78	32.90	78.72	60.84	**73.93**	58.55	43.12
CoverKG	**71.47**	**37.25**	**74.07**	**33.72**	**79.00**	**61.72**	72.46	**59.09**	**45.14**

Table 4: Reward ablations on GraphRAG (token-level F1), with all other training settings fixed.

Variant	NQ	PopQA	HotpotQA	2Wiki	MuSiQue
CoverKG	**13.73**	**20.05**	**14.72**	**12.28**	**8.80**
w/o Compactness	13.27	19.12	14.44	11.97	8.65
w/o Coverage	12.27	18.32	13.79	11.56	8.38

Serialized storage footprint. To quantify the end-to-end deployment cost, Table 7 in Appendix D reports the serialized file size of each deduplicated graph index across the four QA corpora. Specifically, we measure the combined size of the entity and relationship Parquet files for GraphRAG, the unified GraphML file for LightRAG, and the serialized Pickle file for HippoRAG2. CoverKG reduces the total storage footprint by 57% under GraphRAG (54.48→23.32 MB), 76% under LightRAG (212.73→50.48 MB), and 26% under HippoRAG2 (158.98→117.93 MB). These reductions compound with the structural compression reported in Figure 3: fewer elements translate directly into smaller on-disk artifacts. Notably, SFT alone yields modest storage savings (11% for GraphRAG, 42% for LightRAG, 11% for HippoRAG2), whereas the CoverKG reward produces substantially larger reductions on every backend, confirming that compactness-aware optimization achieves storage efficiency beyond what format-level supervision provides.

Critically, this compression is achieved without sacrificing downstream quality. Tables 1 and 3 show that F1 and generation metrics improve simultaneously, confirming that the pruned elements were redundant rather than informative. Unlike the post-hoc canonicalization strategies discussed in §2, the compactness objective achieves this compression at training time: the extractor learns to emit fewer, higher-utility elements during construction itself.

6 Conclusion

We introduced CoverKG, a framework that provides a construction-time training signal for LLM-based graph extraction in knowledge-graph-based RAG pipelines. By decomposing index quality into coverage and compactness and computing a per-chunk reward from their harmonic combination, CoverKG enables reinforcement learning to optimize the extractor without relying on downstream QA accuracy or gold graphs. Experiments across three structurally distinct backends and five QA benchmarks confirm that CoverKG improves accuracy on 12 of 15 configurations (up to $+8.97$ F1) while reducing graph nodes by 48% and edges by 70% on average, and domain-level evaluation on GraphRAG-Bench further demonstrates simultaneous gains in generation quality and graph compression. Future work can further explore this formulation not only as a training objective, but also as a general benchmark for evaluating graph quality.

AI use statement

The authors used AI solely for language polishing and improving readability. No AI tool was used for research design, data analysis, content generation, or figure creation. The authors reviewed and edited all AI-assisted text and take full responsibility for the accuracy, integrity, and originality of the content of this publication.

Reproducibility statement

We support reproducibility by documenting the reward formulation and training procedure in Sections 4.2 and 4.3, and by reporting the datasets, model choices, reward hyperparameters, and evaluation metrics in Section 5.1. Appendix A details marginal computation and the GRPO objective. Appendix B provides the training hyperparameters, compute environment, and backend configurations, while Appendix E provides a worked qualitative example. The public datasets, models, and evaluation frameworks used in the experiments are identified and cited in the paper.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Michele Conforti and Gérard Cornuéjols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the Rado–Edmonds theorem. *Discrete Applied Mathematics*, 7(3):251–274, 1984. doi: 10.1016/0166-218X(84)90003-9.

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. *arXiv preprint arXiv:2404.16130*, 2024. doi: 10.48550/arXiv.2404.16130.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 10746–10761, Suzhou, China, November 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.568. URL https://aclanthology.org/2025.findings-emnlp.568/.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515. PMLR, 2025. URL https://proceedings.mlr.press/v267/gutierrez25a.html.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580.

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.

Haoyu Huang, Jiaxin Bai, Shujie Liu, Yang Wei, Huihao Jing, Hong Ting Tsang, Yisen Gao, Zhongwei Xie, Yufei Li, and Yangqiu Song. DeepRefine: Agentic knowledge refinement via reinforcement learning, 2026. URL https://arxiv.org/abs/2605.10488v2.

Rishabh Iyer and Jeffrey Bilmes. A memoization framework for scaling submodular optimization to large scale problems. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 2340–2349. PMLR, 2019. URL https://proceedings.mlr.press/v89/iyer19b.html.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. doi: 10.1162/tacl_a_00276.

Yassir Lairgi, Ludovic Moncla, Rémy Cazabet, Khalid Benabdeslem, and Pierre Cléau. iText2KG: Incremental knowledge graphs construction using large language models. In *Web Information Systems Engineering – WISE 2024*, volume 15439 of *Lecture Notes in Computer Science*, pp. 214–229, Singapore, 2025. Springer Nature Singapore. doi: 10.1007/978-981-96-0573-6_16. URL https://doi.org/10.1007/978-981-96-0573-6_16.

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, pp. 9459–9474, 2020.

Jing Li, Zhijie Sun, Zhicheng Zhou, Suming Qiu, Junjie Huang, Haijia Sun, and Linyuan Qiu. Agentic-KGR: Co-evolutionary knowledge graph construction through multi-agent reinforcement learning. *arXiv preprint arXiv:2510.09156*, 2025. doi: 10.48550/arXiv.2510.09156.

Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 510–520, Portland, Oregon, USA, 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-1052/.

Haoran Luo, Haihong E, Guanting Chen, Qika Lin, Yikai Guo, Fangzhi Xu, Zemin Kuang, Meina Song, Xiaobao Wu, Yifan Zhu, and Anh Tuan Luu. Graph-R1: Towards agentic GraphRAG framework via end-to-end reinforcement learning. *arXiv preprint arXiv:2507.21892*, 2025. doi: 10.48550/arXiv.2507.21892.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 9802–9822, 2023. doi: 10.18653/v1/2023.acl-long.546.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, January 1979. doi: 10.1109/TSMC.1979.4310076.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016. doi: 10.18653/v1/D16-1264.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. doi: 10.48550/arXiv.2402.03300.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475.

Hong Ting Tsang, Jiaxin Bai, Haoyu Huang, Qiao Xiao, Tianshi Zheng, Baixuan Xu, Shujie Liu, and Yangqiu Song. AutoGraph-R1: End-to-end reinforcement learning for knowledge graph construction. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23351–23374, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.acl-long.1070. URL https://aclanthology.org/2026.acl-long.1070/.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. CESI: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference*, pp. 1317–1327, 2018.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2026, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/6c9e01d6cefbbf4cdd265032550e767f-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. doi: 10.48550/arXiv.2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Chuanyue Yu, Kuo Zhao, Yuhan Li, Heng Chang, Mingjian Feng, Xiangzhe Jiang, Yufei Sun, Jia Li, Yuzhi Zhang, Qingyun Sun, Jianxin Li, and Ziwei Zhang. GraphRAG-R1: Graph retrieval-augmented generation with process-constrained reinforcement learning. In *Proceedings of the ACM Web Conference 2026*, pp. 1398–1409, New York, NY, USA, 2026. Association for Computing Machinery. doi: 10.1145/3774904.3792589. URL https://doi.org/10.1145/3774904.3792589.

Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. doi: 10.48550/arXiv.2506.05176. URL https://arxiv.org/abs/2506.05176.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):article 58, 2024. doi: 10.1007/s11280-024-01297-w.

A Additional Method Details

A.1 Efficient Marginal Computation

A naive evaluation of all n leave-one-out marginals costs $O(mn^2)$ over m source sentences. The $\max$ structure of Eq. (2) admits an exact one-pass computation.

Proposition 1 (Exact $O(mn)$ marginals)**.** *All n marginals Δ_j are computed exactly in $O(mn)$ time and $O(m+n)$ auxiliary space. For each element e_j, its marginal contribution is the weighted decrease in sentence-level coverage caused by removing that element:*

$$\begin{aligned}\Delta_j &= \Gamma(E_c) - \Gamma(E_c \setminus \{e_j\}) \\ &= \sum_{i=1}^{m} w_i \left[\phi(s_i; E_c) - \phi(s_i; E_c \setminus \{e_j\})\right].\end{aligned} \tag{8}$$

A per-sentence term can be nonzero only when removing e_j changes its strongest alignment. Recording each sentence's top-two thresholded alignments therefore evaluates all marginals in a single pass over the similarity matrix.

This form is directly interpretable: an element is useful only to the extent that its removal lowers source-sentence coverage, so merely matching coverage already supplied by another element contributes nothing.

A.2 GRPO Objective

We detail the policy optimization procedure summarized in §4.3. GRPO (Shao et al., 2024) replaces a learned value network with group relative baselines. For each chunk c, we sample a group of G candidate extractions $\{E_c^{(1)}, \ldots, E_c^{(G)}\}$ from $\pi_{\theta_{\text{old}}}$, score each with Eq. (7), and normalize within the group to obtain advantages:

$$\hat{A}^{(g)} = \frac{R\left(c, E_c^{(g)}\right) - \text{mean}_{g'} R\left(c, E_c^{(g')}\right)}{\text{std}_{g'} R\left(c, E_c^{(g')}\right) + \delta_A}. \tag{9}$$

Here, $\delta_A > 0$ is a numerical stabilizer for advantage normalization. The reward is defined at the sequence level, so all tokens of a candidate share its advantage. Let $o^{(g)}$ denote the token sequence of the g-th candidate. The per-token importance ratio is

$$r_t^{(g)}(\theta) = \pi_\theta(o_t^{(g)} \mid c, o_{<t}^{(g)}) / \pi_{\theta_{\text{old}}}(o_t^{(g)} \mid c, o_{<t}^{(g)}), \tag{10}$$

and CoverKG minimizes the clipped GRPO objective:

$$\begin{aligned}\mathcal{L}(\theta) = \; - \; \mathbb{E}_{c, \{E_c^{(g)}\}} \Bigg[\frac{1}{G} \sum_{g=1}^{G} \frac{1}{|o^{(g)}|} \sum_t \min\big(r_t^{(g)} \hat{A}^{(g)}, \\ \text{clip}(r_t^{(g)}, 1-\varepsilon, 1+\varepsilon)\, \hat{A}^{(g)} \big) \Bigg] \\ + \; \lambda_{\text{KL}}\, D_{\text{KL}}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big),\end{aligned} \tag{11}$$

where ε is the clipping range and λ_{KL} penalizes drift from the reference policy π_{ref} (the SFT initialization). The diagnostic components are locally interpretable, while optimization uses their sequence-level aggregate. Training settings are reported in Appendix B.1.

B Additional Experimental Setup

B.1 Training Configuration

We train the Qwen3-8B extractor with LoRA adapters (Hu et al., 2022) in both stages. SFT provides a structurally valid initialization for GRPO. Table 5 lists the optimization hyperparameters and compute environment; training data, model choices, and reward hyperparameters are described in §5.1.

Table 5: Training hyperparameters and compute environment.

Stage	Setting	Value
SFT	LoRA rank	16
	LoRA α	32
	Learning rate	5.0×10^{-5}
	Effective batch size	256
	Training epochs	2
GRPO	LoRA rank	64
	Training batch size	128
	Mini-batch size	64
	Learning rate	1.0×10^{-5}
	Rollout group size G	16
Both	GPUs	$8\times$ NVIDIA A100-SXM4-80GB

B.2 GRAPH-RAG BACKEND CONFIGURATIONS

Table 6 presents YAML-style representations of the configurations used for index construction and retrieval with GraphRAG, LightRAG, and HippoRAG2. All three backends use deterministic LLM decoding (temperature 0) and, where preprocessing is performed by the backend, chunks of 1,200 tokens with a 100-token overlap. The value `null` denotes the backend's built-in prompt, and parameters not listed retain their upstream defaults.

Table 6: Backend configurations used in the experiments, represented in YAML format.

```yaml
graphrag:
  temperature: 0  # Deterministic decoding.

  # ========== Indexing ==========
  build:
    # Chunking
    chunking:
      type: tokens  # tokens/sentences/
     paragraphs
      size: 1200  # Tokens per chunk.
      overlap: 100  # Overlapping tokens.
      encoding_model: o200k_base

    # Entity and relationship extraction
    extract_graph:
      completion_model_id:
     default_completion_model
      prompt: null  # Use the built-in
     prompt.
      max_tokens: 4096
      entity_types:
        - organization
        - person
        - geo
        - event
      max_gleanings: 0

    # Entity-description summarization
    summarize_descriptions:
      completion_model_id:
     default_completion_model
      prompt: null  # Use the built-in
     prompt.
      max_length: 500
      max_tokens: 500
      max_input_tokens: 4000

    # Text embedding
    embed_text:
      embedding_model_id:
     default_embedding_model
      batch_size: 16
      batch_max_tokens: 8191
      names:
        - entity_description

  # ========== Querying ==========
  query:
    mode: local  # local/global

    # Local search
    local_search:
      completion_model_id:
     default_completion_model
      embedding_model_id:
     default_embedding_model
      prompt: null  # Use the built-in
     prompt.
      text_unit_prop: 0.5
      community_prop: 0.0
      conversation_history_max_turns: 5
      top_k_entities: 10
      top_k_relationships: 10
      max_context_tokens: 12000
```

```yaml
lightrag:
  temperature: 0  # Deterministic decoding.

  # ========== Indexing ==========
  build:
    chunk_token_size: 1200
    chunk_overlap_token_size: 100
    entity_extract_max_gleaning: 0
    entity_extract_max_tokens: 4096
    summary_output_max_tokens: 600
    max_parallel_insert: 25
    staged_merge_checkpoint_size: 200
    top_k: 40
    chunk_top_k: 20
    addon_params: {}

  # ========== Querying ==========
  query:
    mode: hybrid  # local/global/hybrid/
     naive/mix
    top_k: 40
    chunk_top_k: 20
    max_entity_tokens: 6000
    max_relation_tokens: 8000
    max_total_tokens: 30000
    response_type: "Single Sentence"

hipporag:
  # ========== Indexing ==========
  build:
    # Preprocessing and chunking
    preprocess_chunk_func: by_token  #
     by_token/by_word
    preprocess_chunk_max_token_size: 1200
    preprocess_chunk_overlap_token_size:
     100

    # Graph construction
    synonymy_edge_topk: 2047
    synonymy_edge_query_batch_size: 1000
    synonymy_edge_key_batch_size: 10000
    synonymy_edge_sim_threshold: 0.8
    graph_edge_batch_size: 50000

    # LLM and PPR
    max_new_tokens: 2048
    temperature: 0
    passage_node_weight: 0.05

  # ========== Querying ==========
  query:
    mode: hipporag  # hipporag/dpr
    linking_top_k: 5
    retrieval_top_k: 200
    damping: 0.5
    qa_top_k: 5
    max_new_tokens: 2048
    temperature: 0
    max_retry_attempts: 5
    batch_size: 20
    num_workers: 20
```

C Analysis by Task Type

We further analyze the results in Table 1 by task regime. CoverKG benefits both task regimes, with mean gains over Base of +2.28 F1 on single-hop tasks and +0.94 F1 on multi-hop tasks across backends. The higher single-hop mean is driven primarily by the +8.97 F1 gain on LightRAG/PopQA, while gains occur in seven of nine multi-hop backend–benchmark cells.

Table 7: Serialized graph index size (MB) compared across backends and training methods on each evaluation corpus. NQ and PopQA share the same underlying corpus 2021 Wikipedia. Lower values indicate less storage overhead.

Method	Extractor	2021Wiki	HotpotQA	2Wiki	MuSiQue	Total
GraphRAG	Base	11.68	14.33	9.47	18.99	54.48
	SFT	12.05	12.82	5.71	17.90	48.48
	CoverKG	5.97	6.23	3.49	7.64	23.32
LightRAG	Base	52.02	59.20	31.46	70.04	212.73
	SFT	46.31	27.73	9.21	39.23	122.49
	CoverKG	15.98	11.81	7.72	14.98	50.48
HippoRAG2	Base	35.89	42.21	31.12	49.76	158.98
	SFT	30.27	38.36	27.66	44.56	140.86
	CoverKG	25.55	31.80	23.66	36.92	117.93

D Detailed Storage Footprint

Table 7 provides the corpus-level storage measurements discussed in §5.5.

E Case Studies

This section presents a qualitative case study illustrating how GRPO training can produce a more compact extracted graph while preserving the information needed for downstream question answering. We select question `qa_466` from the NQ dataset, processed by LightRAG. The question is grounded in the Wikipedia article "New Jersey Drive", represented by a single chunk (64 tokens, 283 characters). The base model `Qwen3-8B` and `Qwen3-8B-GRPO` receive the same chunk.

E.1 Question

Question: what is the movie new jersey drive about
Gold answer: joy riding black teenagers in 1990s Newark, New Jersey

E.2 Source Text

The following chunk is the source text associated with the question above in the NQ corpus:

> New Jersey Drive is a 1995 crime drama film about joy riding black teenagers in 1990s Newark, New Jersey, then known as the "car theft capital of the world". The film was written and directed by Nick Gomez. The executive producer was Spike Lee. Roger Ebert gave the film three stars.

E.3 Extraction Results

Tables 8 and 9 compare the entities and relationships extracted from the same source chunk by the two models. The base model produces 5 entities and 4 relationships, while Qwen3-8B-GRPO produces 4 entities and 2 relationships.

Importantly, the reduction in the size of the graph does not remove the evidence required to answer the question. The Base, SFT, and CoverKG all extract nodes containing the answer-relevant information about *New Jersey Drive*.

E.4 Retrieval Context

We next examine whether the differences in graph extraction affect the context retrieved for downstream answer generation. Table 10 shows the top-ranked entities and chunks returned by LightRAG for `qa_466`. For additional context, we also report retrieval results from SFT.

Table 8: Entities extracted by LightRAG from the same chunk under the base and Qwen3-8B-GRPO. Rows shaded in gray are present only in the base model's output.

#	Entity Name	Type	Description
Qwen3-8B (base model) — 5 entities			
1	New Jersey Drive	Content	1995 crime drama film about joy riding black teenagers in 1990s Newark
2	Nick Gomez	Person	Writer and director of the film
3	Spike Lee	Person	Executive producer of the film
4	Roger Ebert	Person	Film critic; gave the film three stars
5	Newark	Location	City in New Jersey; "car theft capital" in the 1990s
Qwen3-8B-GRPO — 4 entities			
1	New Jersey Drive	Content	1995 crime drama film about joy riding black teenagers in 1990s Newark
2	Nick Gomez	Person	Wrote and directed the film
3	Spike Lee	Person	Executive producer of the film
4	Roger Ebert	Person	Gave the film three stars

Table 9: Relationships extracted by LightRAG from the same source chunk. Rows shaded in gray are present only in the base model's output.

#	Source	Target	Keyword	Description
Qwen3-8B (base model) — 4 relationships				
1	New Jersey Drive	Nick Gomez	written_by, directed_by	Written and directed by Gomez
2	New Jersey Drive	Spike Lee	executive_produced_by	Executive produced by Lee
3	New Jersey Drive	Roger Ebert	reviewed_by	Reviewed by Ebert
4	Newark	New Jersey Drive	set_in	Set in Newark, New Jersey
Qwen3-8B-GRPO — 2 relationships				
1	Nick Gomez	New Jersey Drive	Writing, Directing	Wrote and directed the film
2	Spike Lee	New Jersey Drive	Executive Production	Executive producer of the film

All three models retrieve some irrelevant content because the query term "drive" matches many entities associated with roads and highways. Although all three models extract evidence-bearing nodes for *New Jersey Drive*, only the graph constructed by Qwen3-8B-GRPO allows this evidence to be retrieved under the same downstream retrieval pipeline.

In contrast, retrieval over the graph constructed by the Base model is dominated by several malformed nodes whose descriptions reproduce fragments of the extraction prompt. For example, the entity *Kashmir* has a description beginning with "the location where the event occurred. The entire output must be written in English…". These noisy nodes occupy high-ranking retrieval positions, while the evidence-bearing *New Jersey Drive* entity is not retrieved.

E.5 Analysis

This case study highlights a downstream consequence of graph extraction quality that is not captured by graph size alone. The benefit of GRPO training is not simply that it produces fewer entities and relationships, but that the resulting graph contains a cleaner set of candidates for subsequent retrieval. Under LightRAG's retrieval pipeline, every extracted node may compete for a limited number of high-ranking positions. Consequently, peripheral or malformed nodes can reduce the visibility of otherwise relevant evidence even when that evidence is present in the graph.

The comparison in `qa_466` illustrates this effect. The graph produced by Qwen3-8B-GRPO provides a retrieval space in which the answer-relevant entity can surface, whereas retrieval over the base-model graph is affected by noisy candidates. Thus, preserving answer-relevant information

Table 10: Top-ranked retrieval results for `qa_466` using LightRAG. "Evidence Retrieved" indicates whether the retrieved context contains evidence supporting the gold answer. Underlined entities are malformed nodes whose descriptions repeat extraction-prompt text.

Extractor	Evidence Retrieved	Retrieved Entities	Retrieved Chunks
Base	No	Kashmir, Entity, Ingle, Greg, event, social media	Metro; County Route 549; West Side Story; Golden Gate Bridge
SFT	No	North Carolina Highway 225; County Route 549; US 9W; aired_on	County Route 549; The Aura; High&Low The Movie; U.S. Route 1
CoverKG	Yes	**New Jersey Drive**; 500 Days of Summer; Airport; Evan Almighty	**New Jersey Drive**; Napoleon Dynamite; The Deer Hunter; Forrest Gump

during extraction is necessary but not sufficient: the organization and quality of the surrounding graph also determine whether that information remains accessible to the downstream retriever.

This observation provides a qualitative explanation for the quantitative trends reported in §5.2. GRPO training encourages a more selective graph representation, reducing unnecessary retrieval competition while retaining information useful for question answering. For this example, the resulting change in retrieval behavior is sufficient to alter the final outcome: Qwen3-8B-GRPO answers the question correctly, while the base model returns "insufficient information". Rather than treating graph compactness as an objective by itself, this case suggests that its value lies in producing a higher-quality retrieval space for downstream reasoning and answer generation.