

RAGU: A Multi-Step GraphRAG Engine with a Compact Domain-Adapted LLM

Mikhail Komarov^{1,*} Ivan Bondarenko^{2,*} Stanislav Shtuka^{1,3} Oleg Sedukhin[†]
Roman Shuvalov¹ Yana Dementyeva² Matvey Solovyov¹ Nikolay O. Nikitin¹

¹ITMO University ²Novosibirsk State University ³Far Eastern Federal University

[†]Independent Researcher *Equal contribution

i.bondarenko@g.nsu.ru

Abstract

Graph retrieval-augmented generation (GraphRAG) enhances large language models with structured knowledge, yet existing systems construct knowledge graphs in a single extraction pass, producing noisy entities and brittle retrieval. RAGU, an open-source modular GraphRAG engine, addresses this by separating extraction from consolidation: entities and relations pass through two-stage typed extraction, DBSCAN-backed deduplication, LLM summarization, and Leiden community detection. A key insight motivates a compact extractor: the skills an in-pipeline LLM needs - comprehension, extraction, reasoning over context - are language skills that grow only weakly with model size, unlike factual world knowledge. Accordingly, we train Meno-Lite-0.1, a 7 B model optimized for language skills, which outperforms Qwen2.5-32B on knowledge-graph construction (+12.5% relative harmonic mean) and matches it on English GraphRAG tasks. On GraphRAG-Bench (Medical), RAGU retrieves the most complete context at every factoid level (evidence recall up to 82.4 vs. ≤ 76.1) and overtakes HippoRAG2 on synthesis tasks; on multi-hop factoid QA, the apparent HippoRAG2 advantage is shown to be largely an answer-format artifact. RAGU is installable via `pip install graph_ragu`, runs on a single GPU, and is released under MIT. The source code is publicly available at <https://github.com/RaguTeam/RAGU>, and the Meno-Lite-0.1 model can be obtained from <https://huggingface.co/bond005/meno-lite-0.1>.

1 Introduction

Retrieval-augmented generation (RAG) grounds large language models (LLMs) in external knowledge (Lewis et al., 2020; Gao et al., 2023). Traditional RAG retrieves flat chunks without capturing cross-document entity relationships. Graph

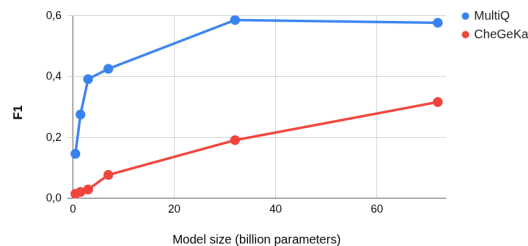


Figure 1: World-knowledge (CheGeKa) vs. language-skill (MultiQ) scaling in the Qwen2.5-Instruct family (F1 on MERA (Fenogenova et al., 2024)): CheGeKa grows $21.1\times$ from 0.5 B to 72 B, MultiQ by only $4\times$ (log-linear slopes 0.65 vs. 0.26).

RAG (GraphRAG) (Edge et al., 2024; Guo et al., 2025; Gutiérrez et al., 2025) addresses this by building a knowledge graph and using graph traversal during retrieval, but practical adoption faces three obstacles.

Obstacle 1: Single-pass extraction. Current systems treat knowledge graph construction as a single LLM extraction pass, producing noisy, duplicated entities with no mechanism to *consolidate* information across chunks.

Obstacle 2: Dependence on expensive LLMs. Extraction quality determines graph quality, so practitioners default to large API models (GPT-4-class). This rests on a premise we argue is unnecessary: the capabilities an LLM needs *inside* a RAG pipeline—comprehension, extraction, reasoning over context—are *language skills*, not factual recall, and language skills grow only weakly with model size. A compact, skill-oriented model therefore suffices.

Obstacle 3: Engineering immaturity. Many open-source frameworks suffer from installation failures or unsafe code paths (e.g., `eval()` on raw LLM output); a GraphRAG engine should be *semantically* strong and *engineerable*: installable, testable, deployable on cost-effective hardware.

Language/World Knowledge Hypothesis. We hypothesize that **world knowledge** scales near-linearly with parameter count, whereas **language skills** scale markedly more slowly. Figure 1 tests this on the Qwen2.5-Instruct family (Yang et al., 2024): CheGeKa (Taktasheva et al., 2022) (world-knowledge quiz) F1 grows $21.1\times$ from 0.5 B to 72 B, while MultiQ (all facts in-context) grows only $4\times$. In a GraphRAG pipeline, the LLM extracts entities, summarizes descriptions, and generates answers from context—all language skills.

We address all three obstacles with two artifacts, released under open licenses:

1. **Meno-Lite-0.1**, a 7 B model fine-tuned from RuadaptQwen2.5-7B (Tikhomirov and Chernyshev, 2025) for RAG-oriented language skills over the NEREL schema (Loukachevitch et al., 2021): it outperforms Qwen2.5-32B on a Russian NEREL-derived KG-construction benchmark (+12.5% harmonic mean) and matches it as an index LLM on GraphRAG-Bench (Appendix B).
2. **RAGU**, a modular multi-step GraphRAG engine whose pipeline separates extraction from consolidation—two-stage typed extraction, DBSCAN-backed summarization, Leiden community detection—yielding cleaner and more connected knowledge graphs.

The two artifacts are complementary. The language-skills hypothesis explains why a 7 B extractor suffices; consolidation explains why the resulting graph retrieves more complete context. Our experiments further indicate that downstream QA quality on GraphRAG-Bench depends more on the pipeline than on extractor scale (Appendix B). RAGU differs from prior systems—Microsoft GraphRAG (Edge et al., 2024), LightRAG (Guo et al., 2025), HippoRAG 2 (Gutiérrez et al., 2025), Wikontic (Chepurova et al., 2026)—by introducing an explicit multi-step consolidation stage and by targeting engineering maturity.

2 System Description

2.1 Multi-Step Graph Construction

RAGU processes documents through six configurable stages (Figure 2):

Step 1: Chunking. Three strategies: *SimpleChunker* (fixed-size overlapping chunks), *SemanticTextChunker* (embedding-based split points), and *SmartSemanticChunker* (cross-encoder reranking).

Step 2: Two-stage extraction. Unlike single-pass systems, RAGU separates entity extraction (Stage 1) from relation extraction (Stage 2). Entities are first extracted and validated against the NEREL schema (29 entity types, 49 relation types), then fed back as constraints: every `source_entity` and `target_entity` in a relation *must* match a validated entity name, eliminating spurious entity–relation mismatches. Optional in-context-learning (ICL) examples, selected by semantic, BM25, hybrid, or random strategies, are injected at both stages; the stage prompts are reproduced in Appendix E.

Step 3: Consolidation. *EntitySummarizer* groups entities by (name, type) and—for entities whose number of duplicate mentions exceeds a configurable threshold (seven by default)—applies LLM summarization; above a second, larger threshold it additionally clusters the descriptions with DBSCAN (eps 0.5, min_samples 2). *RelationSummarizer* follows the same pattern. Reducing noise *before* community detection produces a cleaner graph. LightRAG, the closest prior system in this respect, concatenates descriptions of duplicate mentions during its single extraction pass, but performs no schema-validated typed consolidation or LLM-driven summarization ahead of retrieval.

Steps 4–6: Community detection, summarization, refinement. Hierarchical Leiden clustering partitions the deduplicated graph; an LLM generates structured community reports (title, summary, findings); pluggable modules (e.g., *RemoveIsolatedNodes*) optionally refine.

2.2 Search Engines

RAGU provides five engines: **LocalSearch** (vector-similarity entity retrieval expanded to relations and chunks), **GlobalSearch** (LLM-rated community summarization), **NaiveSearch** (standard vector RAG), **MixSearch** (parallel multi-engine), and **QueryPlanEngine** (DAG decomposition), all with cross-encoder reranking and hybrid dense+sparse retrieval via Qdrant.

2.3 Engineering and Deployment

RAGU is a production-ready Python package (see Appendix A for a comparison with HippoRAG 2): (i) a three-tier storage abstraction (graph/KV/vector) with lifecycle callbacks enables backend swapping; (ii) an async-first API with

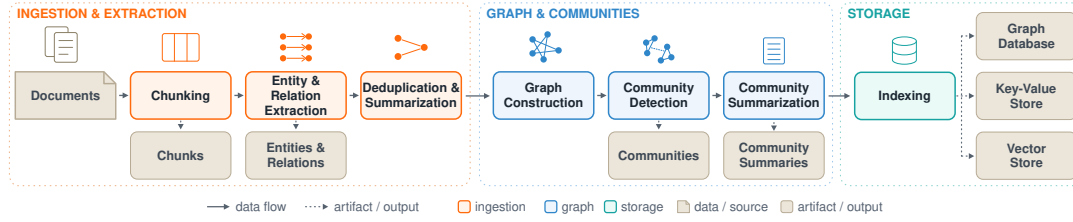


Figure 2: End-to-end indexing pipeline: chunking, NEREL-schema extraction, deduplication and summarization, Leiden communities; all artifacts persist across three swappable storage tiers. Solid arrows: data flow; dashed: artifacts produced at each stage.

bounded concurrency provides production-safe throughput; (iii) structured LLM outputs are validated through Pydantic v2, removing manual JSON post-processing and preventing code injection; (iv) incremental upsert/update/delete with deterministic hash-based IDs, merge policies, and a cross-store consistency auditor. ~ 374 tests and a deterministic mock LLM server enable CI without API keys.

2.4 Compact Model: Meno-Lite-0.1

Meno-Lite-0.1 (Bondarenko et al., 2026) is derived from RuadaptQwen2.5-7B-Lite-Beta (Tikhomirov and Chernyshev, 2025) through continued pretraining (1.3B tokens, Russian+English educational/scientific texts) and supervised fine-tuning (50M tokens) mixing third-party instruction datasets ($\approx 89\%$: MTRAGEval, Russian RAG, tool-calling, multi-hop QA, text simplification), synthetic KG-construction data from LightRAG/RuREBus pipelines ($\approx 7\%$), GPT-4o-mini-distilled instructions ($\approx 4\%$), and a NEREL-instruct subsample ($< 1\%$). The critical distinction from general-purpose LLMs is that its instructions teach the model to *use context* rather than *recall facts*—investing compute in language skills, not world knowledge. Key properties: 128K context window (passkey retrieval 0.98 at 128K) and 47% higher tokenizer efficiency than vanilla Qwen2.5 on Russian text (3.77 vs. 2.57 characters per token). Deployment is single-consumer-GPU: bf16 weights require ≈ 15 GB and the full 128K context adds ≈ 7 GB of paged KV-cache under vLLM (Kwon et al., 2023) (GQA), so a 24 GB consumer GPU hosts the full window. Officially released GGUF quantizations (Q8_0–IQ3_M; 8.1–3.5 GB, importance-matrix calibration) extend deployment to llama.cpp and CPU-only setups: Q5_K_M at ≈ 5.4 GB leaves room for a 128K KV-cache on a 16 GB card, with perplexity correlation

to fp16 of 98.64–99.97% reported in the model card.¹ All experiments use fp16/bf16 weights, and indexing prompts are chunk-sized, so the GraphRAG-Bench construction runs (§3.1) fit on V100-class GPUs.

Training details. Both stages use ms-swift 2.6.1 (?) with bf16 weights on NVIDIA A100-80GB GPUs (≈ 492 GPU-hours in total). Continued pretraining (1.3B tokens; learning rate 3×10^{-5} , 2 epochs) mixes Russian FinePDFs-Edu (570M tokens), English FineWeb-Edu (437M), RuREBus (180M), and an embedding-selected RuLM subset (≈ 113 M). Supervised fine-tuning (50M tokens; learning rate 3×10^{-6} , 3 epochs) uses the four-way instruction mix described above. The full configuration (schedules, batching, sequence lengths, optimizers) is documented in the public model card at <https://huggingface.co/bond005/meno-lite-0.1>.

3 Evaluation

3.1 Setup

We evaluate on four GraphRAG benchmarks: *GraphRAG-Bench* (Xiang et al., 2026) (Medical domain; four difficulty levels: fact retrieval, complex reasoning, contextual summarize, creative generation), *BioASQ* (Krithara et al., 2023), *MuSiQue* (Trivedi et al., 2022), and *2WikiMulti-HopQA* (Ho et al., 2020), with dataset statistics in Appendix D; these cover both graph-centric factoid and synthesis QA and open-domain multi-hop factoid QA. All systems use the *same* answer-generation LLM (gpt-4o-mini), isolating graph construction quality. We compare against LightRAG and HippoRAG 2—the two strongest systems on the GraphRAG-Bench Medical split. We do not re-run Microsoft GraphRAG (Edge

¹<https://huggingface.co/bond005/meno-lite-0.1-gguf>

et al., 2024): its global indexing costs $\sim 40k$ tokens per document (an order-of-magnitude estimate; Appendix C), and the GraphRAG-Bench authors report it trailing LightRAG throughout the Medical split. We vary the index LLM as the independent variable: Meno-Lite-0.1 (7 B) and gpt-oss-20b or Qwen2.5-7B as the second index LLM. Metrics follow the GraphRAG-Bench protocol (Xiang et al., 2026). Answer Correctness (AC) combines statement-level factual correctness against the reference answer with embedding similarity (0.75 factual, 0.25 similarity); Coverage is the fraction of reference evidence points covered by the generated answer; Faithfulness is the fraction of answer claims supported by the retrieved context; Evidence Recall (ER) is the fraction of reference claims supported by the retrieved context; and Context Relevancy measures question–context semantic similarity. The protocol applies ROUGE-L and AC to the factoid levels, Coverage to Contextual Summarize, and Faithfulness to Creative Generation, hence the differing columns of Table 1. LLM-as-judge evaluations use google/gemini-3-flash-preview, ensuring no evaluator–generator overlap.

All experiments use a fixed random seed (42); reported results are single runs. Per-question variability on the multi-hop benchmarks is large (population std up to 0.38 AC over $n \approx 1000$ questions per system), so differences of a few percentage points are best read as noise. GraphRAG-Bench runs used Tesla V100 GPUs with fp16 weights; the remaining benchmarks used Tesla A100-80GB with bf16. Embedders follow the corpora—bge-large-en-v1.5 for the English-only medical corpus, gte-multilingual-base for the multi-hop corpora—and the second index LLM reflects hardware availability (the V100’s memory admits only 7 B-class models, whereas A100-80GB additionally admits gpt-oss-20b; Qwen2.5-7B also keeps the scaling ablation of Appendix B within a single model family). Within each table, the embedder, generator, and judge are identical across all systems, so cross-system comparisons are unaffected.

3.2 GraphRAG-Bench Results

Together, Table 1 and Figure 3 reveal a cross-over. On the two *factoid* levels, HippoRAG 2 leads: personalized PageRank pins down single facts (Fact Retrieval AC 72.4 vs. 54.2 for RAGU, $\Delta = -18.2$ pp). As tasks demand broad synthesis rather than chain-following, the gap closes

monotonically—Complex Reasoning (-14.7 pp), Contextual Summarize (-0.9 pp, parity)—and *reverses* on Creative Generation, where RAGU leads most decisively on Coverage (57.4 vs. 34.7) and on Faithfulness (34.2 vs. 26.6; the spread between HippoRAG 2’s own index LLMs indicates the single-run noise floor on this metric), while its AC edge (59.0 vs. 56.9) stays within that noise floor. LightRAG is weakest on every generation metric, despite leading Evidence Recall on Creative Generation; this pattern is consistent with the hypothesis that single-pass free-form extraction produces a structurally poorer graph than typed multi-step consolidation.

Retrieval results support the mechanism: RAGU attains the highest Evidence Recall on all three factoid and summarization levels (82.4, 74.5, 74.8 vs. at most 76.1, 71.3, 71.8) and the highest Context Relevancy on the same levels (89.5, 79.2, 85.0 vs. at most 84.9, 72.5, 79.6). The sole exception is Creative Generation—the only non-factoid level—where LightRAG leads ER (59.9 vs. 53.1) and HippoRAG 2 drops sharply (36.2): personalized PageRank confines retrieval to the neighbourhood of query nodes, sufficing for chain-following but starving synthesis queries of broader evidence. On the generation side the picture reverses: RAGU wins every metric at that level—AC, Coverage, Faithfulness (Table 1). The divergence between retrieval and generation metrics has a single explanation: Evidence Recall measures the completeness of retrieved context, not its precision or its usability by the generator. High recall without precision (LightRAG on Creative Generation) does not translate into generation quality, since it retrieves broad context but generates terse or over-general answers that cover almost none of the reference evidence points (Coverage 3–4), whereas precise but narrow retrieval (HippoRAG 2 on Fact Retrieval) suffices for single-fact answers. RAGU is the only system that leads both Evidence Recall and Context Relevancy on the three factoid and summarization levels. A no-graph baseline sharpens the cross-over: NaiveRAG (flat vector retrieval, Table 1) roughly matches RAGU on factoid AC (55.1 vs. 54.2) but trails it by 9–16 pp on Contextual Summarize and Creative Generation, across AC, Coverage, and Faithfulness. The graph’s benefit is therefore concentrated on summarization and synthesis, not on single-fact QA. Ablations (Appendix B) show that the ICL and validation toggles each shift AC by at most ~ 1.5 pp, so the gap to competitors reflects

Table 1: Generation quality on GraphRAG-Bench (Medical). AC=Answer Correctness, Cov=Coverage, Faith=Faithfulness ($\times 100$; definitions in §3.1). All systems share bge-large-en-v1.5 and gpt-4o-mini; only the index LLM varies. RAGU rows use ICL = 1, Val=yes. NaiveRAG is RAGU’s NaiveSearchEngine (flat vector RAG over chunks; no graph construction); its two rows are two runs of the same baseline. LightRAG’s Creative Generation Coverage of 3–4 is genuine: its answers cover almost none of the reference evidence points. Best per column in **bold**.

System	Index LLM	Fact Retr.	Complex Reas.	Contextual Summ.		Creative Gen.		
		AC	AC	AC	Cov	AC	Cov	Faith
LightRAG	Qwen2.5-7B	25.9	20.3	22.1	51.0	14.2	3.1	23.1
LightRAG	Meno-Lite-0.1	26.2	20.2	22.6	51.2	14.4	3.9	27.6
HippoRAG 2	Qwen2.5-7B	72.7	67.8	64.6	51.6	57.2	33.2	31.5
HippoRAG 2	Meno-Lite-0.1	72.4	68.4	65.0	51.7	56.9	34.7	26.6
RAGU	Qwen2.5-7B	54.1	54.6	64.9	73.2	58.1	56.4	35.1
RAGU	Meno-Lite-0.1	54.2	53.7	64.1	71.1	59.0	57.4	34.2
NaiveRAG	Qwen2.5-7B	55.0	55.7	53.3	63.6	48.9	44.8	26.6
NaiveRAG	Meno-Lite-0.1	55.1	54.8	53.6	61.7	48.6	41.8	24.0

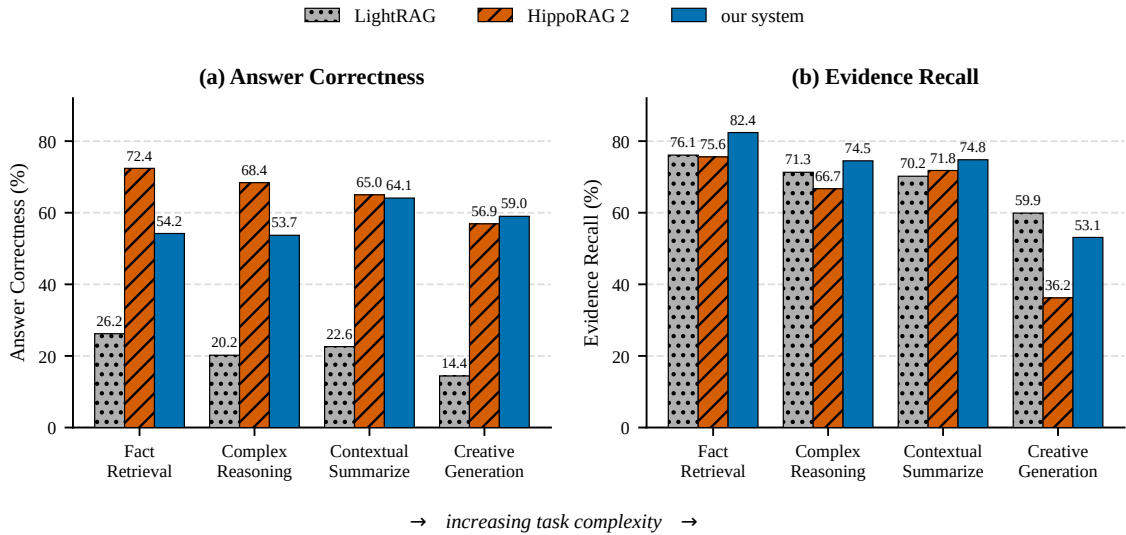


Figure 3: Cross-over by task complexity on GraphRAG-Bench (Medical); all systems use Meno-Lite-0.1 (7 B) for indexing and gpt-4o-mini for generation. (a) Answer Correctness: RAGU trails HippoRAG 2 on Fact Retrieval but leads on Creative Generation. (b) Evidence Recall: RAGU is highest on the three factoid and summarization levels; LightRAG leads on Creative Generation. Difficulty increases left to right.

the combined effect of RAGU’s structural choices, which we do not isolate component-wise; an ablation disabling consolidation while holding chunking, extraction, and retrieval fixed is the most direct test of this mechanism and is left to future work. Across extraction LLMs from 3 B to 32 B, end-to-end AC varies by at most ~ 1.5 pp, suggesting that QA quality on GraphRAG-Bench Medical is largely insensitive to extractor scale once consolidation is applied—even though standalone extraction quality is not (Table 3).

3.3 Multi-Hop QA Results

These benchmarks are pure factoid QA with short gold answers, where answer *format* strongly affects overlap metrics. We therefore report two generation protocols (Table 2): (a) each system’s default verbose prompt, and (b) a terse prompt forcing a single direct answer (both in Appendix E). HippoRAG 2’s default already emits terse output, making it a format anchor; the small differences between its panels (a) and (b) stem from re-generation stochasticity, not from a format change. NaiveRAG is RAGU’s NaiveSearchEngine and shares our generation prompt.

Table 2: Multi-hop QA under two generation protocols (AC = Answer Correctness, RL = ROUGE-L, $\times 100$). All systems share gte-multilingual-base and gpt-4o-mini; only the index LLM varies (GPT = gpt-oss-20b, Ours = Meno-Lite-0.1). (a) Verbose prompts (each system’s default); (b) terse prompts (both in Appendix E); HippoRAG 2’s default is already terse. NaiveRAG is RAGU’s NaiveSearchEngine and shares our generation prompt. Best AC per column in **bold**.

System	BioASQ		MuSiQue		2WikiMultiHop	
	AC	RL	AC	RL	AC	RL
<i>(a) Verbose generation prompt</i>						
NaiveRAG	55.3	12.4	41.7	6.9	43.5	12.0
LightRAG (GPT)	43.9	6.5	34.5	3.8	36.2	8.2
HippoRAG 2 (GPT)	74.1	48.8	56.3	42.7	65.9	54.7
RAGU (GPT)	56.0	12.2	43.5	7.6	46.6	13.2
RAGU (Ours)	54.5	12.1	42.0	7.4	45.2	12.9
<i>(b) Terse generation prompt</i>						
NaiveRAG	71.7	49.2	36.6	24.7	53.7	45.2
LightRAG (GPT)	63.9	42.9	26.0	12.1	44.3	35.7
HippoRAG 2 (GPT)	72.4	48.8	54.4	42.6	63.5	54.7
RAGU (GPT)	72.9	48.7	40.1	26.5	58.0	49.6
RAGU (Ours)	72.8	48.2	40.7	27.5	55.1	46.3

Under the verbose protocol (a), HippoRAG 2 dominates every column (AC up to 74.1 vs. 56.0 for RAGU on BioASQ). Part of this gap is a format artifact: verbose answers mismatch the terse gold references, depressing both ROUGE-L (12 vs. 49) and Answer Correctness. Once format is controlled (b), the gap narrows but does not vanish: RAGU slightly exceeds HippoRAG 2 on BioASQ AC (72.9 vs. 72.4) and closes the 2Wiki-MultiHopQA gap from -19.3 to -5.5 pp (58.0 vs. 63.5), while HippoRAG 2 retains a genuine lead on MuSiQue (54.4 vs. 40.1), the hardest benchmark, where its personalized PageRank follows reasoning chains that consolidated retrieval does not surface—and where terseness *lowers* Answer Correctness for every system, including HippoRAG 2 itself.

The refined picture is one of complementary strengths: HippoRAG 2 excels at chain-following multi-hop and single-fact precision, while RAGU is competitive on factoid QA once format is controlled and superior on context-breadth tasks—with a locally served 7 B extractor (Meno-Lite-0.1) versus a 20 B gpt-oss-20b; the 1–2 pp gap between RAGU’s two index LLMs supports Meno-Lite-0.1 as a drop-in replacement within single-run variability.

3.4 Model Evaluation

Table 3 is consistent with the prediction: Meno-Lite-0.1 achieves the highest harmonic mean on our Russian NEREL-derived IE benchmark (Bon-

Table 3: IE benchmark (KG construction): NER = entity recognition (F1), RE = relation extraction (F1), Def/RDef = entity/relation definition (chrF++), HM = harmonic mean of all four. Qwen3 rows are Instruct-2507 checkpoints; Qwen3-30B-A3B is MoE with 3 B active parameters. All rows are our own runs under the LM Evaluation Harness (Gao et al., 2024).

Model	Size	NER	Def	RE	RDef	HM
Meno-Lite-0.1	7B	0.504	0.527	0.347	0.558	0.468
Qwen2.5-32B	32B	0.536	0.528	0.239	0.599	0.416
gemma-3-27b	27B	0.544	0.482	0.224	0.583	0.396
Qwen3-30B-A3B	30B	0.490	0.408	0.262	0.558	0.396
Qwen2.5-14B	14B	0.510	0.518	0.222	0.583	0.396
gemma-3-4b	4B	0.447	0.396	0.236	0.511	0.365
Qwen2.5-7B	7B	0.477	0.479	0.192	0.541	0.356
Qwen3-4B	4B	0.503	0.436	0.181	0.561	0.345
T-lite-1.0	7B	0.466	0.464	0.174	0.533	0.336

darenko, 2026a) (see Limitations for the fine-tuning overlap), outperforming Qwen2.5-32B by 12.5% relative, driven by relation extraction (F1 0.347 vs. 0.239)—the sub-task most dependent on language comprehension. Recent 4B-class checkpoints (gemma-3-4b, Qwen3-4B-Instruct-2507) trail Meno-Lite-0.1 by 0.10–0.12 in harmonic mean. On MERA (Fenogenova et al., 2024), Meno-Lite-0.1 scores 0.555 overall with near-perfect passkey retrieval (0.98 at 128 K tokens, LIBRA (Churin et al., 2025)), indicating robust long-context handling.

Fine-tuning payoff vs. pipeline robustness.

Meno-Lite-0.1’s standalone extraction advantage (Table 3) compresses to ≤ 1 pp on end-to-end GraphRAG-Bench QA, in every pipeline we tested (HippoRAG 2, LightRAG, and our own); a dedicated Qwen2.5-32B index run shifts AC by at most 0.9 pp across all four levels (Appendix B). Two readings are consistent with this. Either consolidation absorbs extractor differences, or factoid medical QA is simply insensitive to graph quality in this range; our data do not separate the two, and the second reading would equally limit the measured benefit of RAGU’s own graph. What the numbers do support is a practical claim: at 7 B, Meno-Lite-0.1 matches 32B-class extraction quality standalone and loses nothing downstream, so nothing is gained by paying for a larger extractor. Extraction quality remains directly consequential where the graph itself is the deliverable (an inspectable, exportable artifact), and there Meno-Lite-0.1 is the strongest model in its size class, most visibly for Russian (0.468 vs. 0.356 over Qwen2.5-7B, Table 3).

Table 4: Entities extracted from the Ritchie passage.

Entity	NEREL Type
Dennis Ritchie	PERSON
Alistair E. Ritchie	PERSON
C Programming Language	PRODUCT
Unix Operating System	PRODUCT
Bell Laboratories	ORGANIZATION
October 12, 2011	DATE
70	AGE
Murray Hill	DISTRICT
New Jersey	STATE_OR_PROV

Table 5: Relations extracted from the Ritchie passage (five of eight shown).

Source	Target	Relation
Dennis Ritchie	C Programming Language	WORKS_AS
Dennis Ritchie	Unix Operating System	WORKS_AS
Dennis Ritchie	October 12, 2011	DATE_OF_DEATH
Alistair E. Ritchie	Dennis Ritchie	PARENT_OF
Bell Laboratories	Murray Hill	LOCATED_IN

4 Demonstration

Case study. We illustrate RAGU’s pipeline on a single sentence about Dennis Ritchie, produced by the bundled example script:

Dennis Ritchie, the creator of the C programming language, and the co-creator of the Unix operating system, died on October 12, 2011, at the age of 70. His father, Alistair E. Ritchie, worked for many years at Bell Laboratories in Murray Hill, New Jersey.

Extraction. The two-stage extractor identifies nine typed entities under the NEREL schema (Table 4). Stage 2 then extracts eight relations, all constrained to this validated entity set (Table 5).²

Community detection. Leiden clustering partitions the 9-entity graph (Figure 4) into two communities: one centred on Dennis Ritchie and his creations (5 entities, 4 relations), the other linking Bell Laboratories, Murray Hill, New Jersey, and Alistair Ritchie through spatial and professional ties (4 entities, 3 relations). An LLM generates a structured summary for each.

Multi-hop retrieval. Using the built graph, RAGU’s LocalSearchEngine answers questions that require traversing multiple edges:

Q: Where did the father of the creator of the C programming language work?
A: Alistair E. Ritchie... worked at **Bell Laboratories**.

²The NEREL schema defines no CREATOR_OF relation; WORKS_AS is the closest available type for “creator of a programming language” and reflects a known coarseness of the ontology rather than an extraction error.

Q: What did the person who died on October 12, 2011, create?

A: Dennis Ritchie... created the **C programming language** and co-created the **Unix operating system**.

Availability. RAGU is installable via `pip install graph_ragu`; full API documentation, examples, and a demonstration video³ are available at <https://github.com/RaguTeam/RAGU>. The system is released under the MIT license; the Meno-Lite-0.1 model is distributed under Apache 2.0 at <https://huggingface.co/bond005/meno-lite-0.1>, with GGUF quantizations at <https://huggingface.co/bond005/meno-lite-0.1-gguf>.

5 Conclusion

We argued that the LLM inside a RAG pipeline needs language skills—not world knowledge—and that these skills scale weakly with model size. Meno-Lite-0.1 delivers 32 B-class extraction at a fraction of the serving cost, and RAGU combines this compact extractor with a modular multi-step pipeline that retrieves the broadest, most usable context on GraphRAG-Bench and leads both HippoRAG 2 and LightRAG on synthesis tasks, while HippoRAG 2 conversely excels at single-fact precision and chain-following multi-hop reasoning. Practically: prefer RAGU when answers must synthesize broad context (summarization, creative generation, long-form QA) under a single-GPU budget, and prefer chain-traversal systems for precise multi-hop fact lookup. Both artifacts are released under open-source licenses. Open problems include a component-wise isolation of consolidation’s contribution, a scalable graph-database backend, and evaluation against newer compact checkpoints.

Acknowledgments

The research was carried out within the state assignment of the Ministry of Science and Higher Education of the Russian Federation (project No. FSER-2024-0004).

We thank Kirill Novgorodtsev for the web frontend (built with the \$mol framework; https://github.com/hyoo-ru/mam_mol) and Yaroslav Svetlov for the backend of the RAGU demo website.⁴

³<https://youtu.be/bicJDMJuQfg>

⁴<https://raguteam.github.io/web>

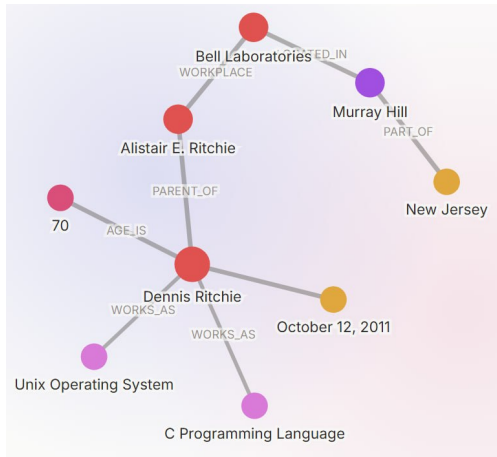


Figure 4: Knowledge graph built from the Ritchie passage. Nodes are typed entities; edges are typed relations. Two communities emerge: Ritchie’s professional legacy and the Bell Labs geographic cluster.

Limitations

First, our scaling evidence rests on a single model family (Qwen2.5) and select tasks; the trend holds across six model sizes, but remains a well-supported hypothesis rather than a general law. Our comparison against compact checkpoints likewise stops at the Qwen3/gemma-3 generation (Table 3); evaluating newer Qwen3.5-class releases remains future work.

Second, Meno-Lite-0.1 trades parametric factual recall for contextual grounding and should not serve as a standalone knowledge base; its multi-hop reasoning degrades beyond 32 K tokens, a limitation typical of 7 B-class models. We also note a distributional-overlap caveat on our IE benchmark (Bondarenko, 2026a): the supervised fine-tuning of Meno-Lite-0.1 uses the train and validation splits of NEREL, whereas the benchmark uses only the held-out test split with different instruction wordings; the overlap is confined to the annotation schema and text domain, not to the benchmark documents—but a residual advantage cannot be fully ruled out.

Finally, RAGU’s default NetworkX graph backend—a swappable BaseGraphStorage adapter—does not by itself scale to massive corpora (millions of nodes), so such deployments require a dedicated graph-database adapter. Final graph quality remains sensitive to the extraction LLM: weak base models introduce structural noise that consolidation cannot fully remove. We further note that our ablations (Appendix B) vary the ICL and validation toggles and the extraction LLM, but do not disable the consolidation stage itself while

holding everything else fixed; the contribution of consolidation is therefore inferred from the RAGU-versus-LightRAG contrast and from the Evidence Recall pattern rather than isolated directly. Microsoft GraphRAG is likewise not benchmarked first-hand: its token-intensive indexing made a full re-run prohibitive, so we rely on the published GraphRAG-Bench evaluation, which places it below LightRAG throughout the Medical split; a first-party replication is left to future work.

Ethics Statement

Data Provenance and Licensing. Meno-Lite-0.1 was trained exclusively on publicly available datasets that are cited in the References—including educational web corpora (FineWeb-Edu), Russian-language academic texts (RuLM), information-extraction datasets (Loukachevitch et al., 2021), multi-hop QA benchmarks (Tang and Yang, 2024; Katsis et al., 2025), and synthetic instructions generated with GPT-4o-mini. No personally identifiable information was included in any training or evaluation corpus. The IE benchmark is a test-only derivative of the human-annotated NEREL corpus (Bondarenko, 2026b) and is released under an MIT license; its integration into the LM Evaluation Harness (Gao et al., 2024) enables standardized, reproducible evaluation.

Environmental Impact and Democratization.

A central design goal of this work is to make GraphRAG accessible beyond large industrial laboratories. Because the Language/World Knowledge Hypothesis (§1) predicts—and our experiments indicate—that a 7 B model suffices for extraction, RAGU+Meno-Lite-0.1 runs on a *single consumer GPU*, rather than the multi-GPU clusters required by frontier models. Token volumes are empirical, and GPU-hour costs are analytic estimates (Appendix C): graph construction processes ~8 k tokens/document at ~2 k tok/s, costing ~\$0.001/doc on rented GPUs—roughly two orders of magnitude less than the ~\$0.10/doc of commercial API-based alternatives, and a factor of a few less than equally local 20B–32B extractors, which require A100-80GB-class serving. At corpus scale (100 k documents), the difference is ~\$100 (local GPU) vs. ~\$10 000 (commercial API). The lower compute footprint translates into lower energy consumption and CO₂ emissions per document indexed. Furthermore, by supporting both English and Russian through `Settings.language` and by training

on Russian-language corpora, RAGU lowers the barrier for non-English-speaking users and under-represented language communities.

Potential Misuse. Like any RAG system, RAGU can amplify biases present in its indexing corpus: if the source documents contain prejudiced or factually incorrect content, the extracted knowledge graph will reflect those biases, and generated answers may inherit them. Meno-Lite-0.1 additionally inherits biases from its pretraining and fine-tuning corpora. We recommend domain-specific evaluation before deployment in sensitive applications (healthcare, legal, news). Meno-Lite-0.1 deliberately trades parametric factual recall for context-grounded skills and should not be used as a standalone knowledge base, reducing the risk of confident hallucination but increasing dependence on corpus quality. On the engineering side, RAGU validates all LLM outputs through Pydantic models rather than executing raw model responses (unlike systems that use `eval()`), which eliminates a class of code-injection attacks from adversarial model output.

Bias and Fairness. The NEREL schema (Loukachevitch et al., 2021) underlying RAGU’s extraction was developed for Russian news text; its entity and relation types reflect that domain. Applying RAGU to other languages or domains may require schema adaptation. The IE benchmark similarly reflects Russian-language text characteristics (e.g., heavy inflection, addressed by Snowball stemming in the metric). Users should be aware of potential performance degradation when operating outside the schema’s design domain.

Transparency and Reproducibility. All code, model weights, and benchmark data are publicly released under open licenses (see §4 for the full availability statement). Every domain object (entity, relation, chunk) carries a deterministic MD5 identifier, allowing any retrieved result to be traced back to its source text. The IE benchmark is integrated into the public LM Evaluation Harness repository under the `nerel-bench` task group, ensuring that any causal LLM can be evaluated under identical conditions.

References

- Ivan Bondarenko. 2026a. Nerel-bench: A benchmark for evaluating llms on russian knowledge graph construction tasks. https://huggingface.co/datasets/bond005/NEREL_instruct. Integrated into the LM Evaluation Harness under the `nerel-bench` task group.
- Ivan Bondarenko. 2026b. Nerel-instruct: An instruction-based dataset for russian information extraction. https://huggingface.co/datasets/bond005/NEREL_instruct.
- Ivan Bondarenko, Roman Derunets, Oleg Sedukhin, Mikhail Komarov, Ivan Chernov, and Mikhail Kulakov. 2026. **RaguTeam at SemEval-2026 task 8: Meno and Friends in a judge-orchestrated LLM ensemble for faithful multi-turn response generation.** In *Proceedings of the 20th International Workshop on Semantic Evaluation (2026)*, pages 1678–1694, San Diego, California, USA. Association for Computational Linguistics.
- Alla Chepurova, Aydar Bulatov, Mikhail Burtsev, and Yuri Kuratov. 2026. **Wikontic: Constructing Wikidata-aligned, ontology-aware knowledge graphs with large language models.** In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8304–8319, Rabat, Morocco. Association for Computational Linguistics.
- Igor Churin, Murat Apishev, Maria Tikhonova, Denis Shevelev, Aydar Bulatov, Yuri Kuratov, Sergei Averkiev, and Alena Fenogenova. 2025. **Long context benchmark for the Russian language.** In *Proceedings of the 6th Workshop on Computational Approaches to Discourse, Context and Document-Level Inferences (CODI 2025)*, pages 1–13, Suzhou, China. Association for Computational Linguistics.
- Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. **From local to global: A graph RAG approach to query-focused summarization.** *arXiv preprint arXiv:2404.16130*.
- Alena Fenogenova, Artem Chervyakov, Nikita Martynov, Anastasia Kozlova, Maria Tikhonova, Albina Akhmetgareeva, Anton Emelyanov, Denis Shevelev, Pavel Lebedev, Leonid Sinev, Ulyana Isaeva, Katerina Kolomeytseva, Daniil Moskovskiy, Elizaveta Goncharova, Nikita Savushkin, Polina Mikhailova, Anastasia Minaeva, Denis Dimitrov, Alexander Panchenko, and Sergey Markov. 2024. **AMERA: A comprehensive LLM evaluation in Russian.** In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9920–9948, Bangkok, Thailand. Association for Computational Linguistics.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff,

- Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. [The language model evaluation harness](#).
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. [Retrieval-augmented generation for large language models: A survey](#). *arXiv preprint arXiv:2312.10997*.
- Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. [LightRAG: Simple and fast retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 10746–10761, Suzhou, China. Association for Computational Linguistics.
- Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. 2025. [From RAG to memory: Non-parametric continual learning for large language models](#). In *Forty-second International Conference on Machine Learning*.
- Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. [Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Yannis Katsis, Sara Rosenthal, Kshitij Fadnis, Chulaka Gunasekara, Young-Suk Lee, Lucian Popa, Vraj Shah, Huaiyu Zhu, Danish Contractor, and Marina Danilevsky. 2025. [mtrag: A multi-turn conversational benchmark for evaluating retrieval-augmented generation systems](#). *Transactions of the Association for Computational Linguistics*, 13:784–808.
- Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. [Bioasqqa: A manually curated corpus for biomedical question answering](#). *Scientific Data*, 10:170.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. [Efficient memory management for large language model serving with PagedAttention](#). In *Proc. 29th ACM Symp. Operating Systems Principles (SOSP)*.
- 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. 2020. [Retrieval-augmented generation for knowledge-intensive NLP tasks](#). In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20*, Red Hook, NY, USA. Curran Associates Inc.
- Natalia Loukachevitch, Ekaterina Artemova, Tatiana Batura, Pavel Braslavski, Ilia Denisov, Vladimir Ivanov, Suresh Manandhar, Alexander Pugachev, and Elena Tutubalina. 2021. [NEREL: A russian dataset with nested named entities, relations and events](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 876–885, Held Online. IN-COMA Ltd.
- OpenAI. 2026. OpenAI API pricing. <https://openai.com/api/pricing/>. Accessed Jun. 2026; gpt-4o \$2.50/\$10 per M input/output tokens, gpt-4o-mini \$0.15/\$0.60 per M.
- Ekaterina Taktasheva, Tatiana Shavrina, Alena Fenogenova, Denis Shevelev, Nadezhda Katricheva, Maria Tikhonova, Albina Akhmetgareeva, Oleg Zinkevich, Anastasiia Bashmakova, Svetlana Iordanskaia, and 1 others. 2022. [Tape: Assessing few-shot russian language understanding](#). In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2472–2497.
- Yixuan Tang and Yi Yang. 2024. [MultiHop-RAG: Benchmarking retrieval-augmented generation for multi-hop queries](#). In *Proceedings of the Conference on Language Modeling (COLM 2024)*.
- Mikhail Tikhomirov and Daniil Chernyshev. 2025. [Ruadapt: Cost-effective large language model lingual adaptation](#). *Doklady Mathematics*, 112. Model checkpoint: <https://huggingface.co/RefalMachine/RuadaptQwen2.5-7B-Lite-Beta>.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. [MuSiQue: Multi-hop questions via single-hop question composition](#). *Transactions of the Association for Computational Linguistics*, 10:539–554.
- Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. 2026. [When to use graphs in RAG: A comprehensive analysis for graph retrieval-augmented generation](#). In *International Conference on Learning Representations (ICLR 2026)*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2024. [Qwen2.5 technical report](#). *arXiv preprint arXiv:2412.15115*.

A Engineering Comparison

Table 6 compares RAGU with HippoRAG 2 along the production risks we encountered while benchmarking open-source GraphRAG frameworks.⁵

Table 6: Engineering comparison between RAGU and HippoRAG 2, organized by the production risk each property addresses.

RAGU	HippoRAG 2
<i>Silent data loss on crash</i>	
Lifecycle callbacks at every storage tier	No shared flush protocol
<i>Backend migration cost</i>	
Three swappable tiers behind one interface	Concrete Parquet/Pickle/SQLite; pipeline rewrite required
<i>API-bound throughput</i>	
Native async with semaphore-bounded batching and rate limits	Synchronous core; thread-pool parallelism
<i>Code execution from LLM output</i>	
Pydantic-validated structured outputs	<code>eval()</code> on raw LLM responses (replaced upstream after d437bfb1)
<i>Transient failure recovery</i>	
tenacity retry; per-chunk error isolation	Broad except; assert False as control flow (replaced upstream after d437bfb1)
<i>Regression detection</i>	
~374 tests; deterministic mock LLM server	<code>tests/</code> added upstream after d437bfb1; demo scripts still require live API keys
<i>Incremental maintenance</i>	
Explicit upsert/update/delete; merge policies; consistency auditor	Hash dedup; file-existence checks
<i>Reproducible deployment</i>	
Loose version constraints; optional GPU extras	Hard-pinned <code>torch+vllm</code> ; CUDA-locked
<i>Modularity</i>	
Thirteen abstract base classes; constructor injection	Monolithic indexing class (~1.6k lines)

HippoRAG 2’s `eval()` on a regex-filtered substring of the raw LLM response is an arbitrary-code-execution surface, and its offline indexing path used to terminate with `assert False`, which `python -O` strips; both issues are fixed in the current upstream, but the pinned snapshot predates those fixes. Conversely, RAGU’s storage abstractions reduce a

⁵All file:line references point to commit d437bfb1 of OSU-NLP-Group/HippoRAG (2025-09-04), so the comparison is reproducible against a fixed snapshot. As of the current upstream main (commit 2f52a86, 2026-08-23), the `assert`-based control flow and `eval()` on raw LLM output have been replaced by an explicit exception and JSON parsing, and a `tests/` directory added; the hard-pinned `torch/transformers` versions remain. Affected rows are annotated.

Table 7: Generation AC ($\times 100$) for selected RAGU configurations on GraphRAG-Bench Medical (ICL = 1, Val = yes unless noted).

ICL	Val	LLM	FR	CR	CS	CG
0	no	Qwen2.5-3B	53.1	53.1	63.8	58.2
0	no	Qwen2.5-7B	54.3	53.4	64.1	58.2
0	no	Qwen2.5-32B	53.5	53.9	64.3	58.1
0	no	Meno-Lite-0.1	54.3	54.0	63.9	58.8
1	no	Qwen2.5-14B	53.9	54.2	65.6	59.1
1	yes	Qwen2.5-7B	54.1	54.6	64.9	58.1
1	yes	Meno-Lite-0.1	54.2	53.7	64.1	59.0

NetworkX/NanoVDB→Neo4j/Qdrant migration to two constructor arguments, and the deterministic mock LLM server turns a full CI run into seconds of CPU time.

Provenance. Key anchors, verifiable against the pinned commit (paths relative to repository root): `eval()` on raw LLM output at `openie_openai.py:36,88`; `assert False` at `HippoRAG.py:216` and `embedding_model/__init__.py:30`; bare `except` at `openie_openai.py:63,112`; the monolithic 1611-line `HippoRAG.py`.

B GraphRAG-Bench Ablation Summary

Table 7 reports generation AC for selected RAGU configurations; full results for all 11 configurations are in the repository.

Key observations: model size (3B–32B) shifts AC by ≤ 1.5 pp; Qwen2.5-32B matches Meno-Lite-0.1 within 0.9 pp; ICL and validation each contribute < 1 pp; Meno-Lite-0.1 and Qwen2.5-7B stay within 1 pp in every configuration.

C Cost Analysis

Table 8 reports graph-construction (indexing) cost, a one-time per-document operation separate from answer generation (all systems share gpt-4o-mini, so query-time cost is a common baseline). For construction, MS-GraphRAG uses a commercial gpt-4o API (\$2.50/M input tokens (OpenAI, 2026)); the remaining systems run a local model under vLLM (Kwon et al., 2023)—LightRAG and HippoRAG 2 use gpt-oss-20b, RAGU uses Meno-Lite-0.1—billed at a fixed GPU-hour rate ($\approx \$1/\text{h}$ at ~ 2 k tok/s, i.e., $\sim \$0.001/\text{doc}$; near zero on owned hardware). Within the local-serving regime, the saving is a factor of a few rather than orders of magnitude: gpt-oss-20b and Qwen2.5-32B require

Table 8: Approximate graph-construction (indexing) token volume and cost per document; query-time generation (gpt-4o-mini, common to all) is omitted. * marks empirical measurements; the unmarked MS-GraphRAG count is an order-of-magnitude estimate. RAGU’s volume is pipeline-determined and identical across index LLMs.

System	Indexing model	Tokens/doc	Cost/doc
MS-GraphRAG (global)	gpt-4o (API)	~40 k	~\$0.10
RAGU+Qwen2.5-32B	Qwen2.5-32B (local)	~8 k*	fixed GPU (A100-80GB)
HippoRAG 2	gpt-oss-20b (local)	~6 k*	fixed GPU (A100-80GB)
LightRAG	gpt-oss-20b (local)	~8 k*	fixed GPU (A100-80GB)
RAGU+Meno-Lite-0.1	Meno-Lite-0.1 (local)	~8 k*	fixed GPU (24 GB-class)

Table 9: Evaluation benchmarks. For GraphRAG-Bench (Medical), questions are broken down by difficulty level (FR = Fact Retrieval, CR = Complex Reasoning, CS = Contextual Summarize, CG = Creative Generation).

Benchmark	Corpus	Questions
GraphRAG-Bench (Medical)	NCCN guidelines	2,062
FR / CR / CS / CG		1,098 / 509 / 289 / 166
BioASQ	biomedical abstracts	500
MuSiQue	Wikipedia (multi-hop)	497
2WikiMultiHopQA	Wikipedia (multi-hop)	999

A100-80GB-class GPUs (market rental $\approx$ \$2–3/h) versus $\approx$ \$1/h for the 24 GB-class GPU that serves a 7 B model. All local figures are analytic estimates rather than metered costs; token volumes are averaged empirical measurements (the MS-GraphRAG figure reflects the token intensity of its global indexing (Edge et al., 2024)), with each model’s own tokenizer, so volumes are not directly comparable across systems, though the order of magnitude is.

D Dataset Statistics

Table 9 summarizes the four evaluation benchmarks. The GraphRAG-Bench Medical corpus is a single concatenated NCCN context ($\approx$ 1.05M characters), chunked by RAGU into 2,848 chunks yielding a consolidated graph of 9,598 entities and 20,901 relations; multi-hop counts are the questions evaluated in our runs.

E Prompt Templates

All prompts are Jinja2 templates (`{{ . . . }}`) placeholders filled at run time; outputs validated against the Pydantic schemas of §2; full versions, validation prompts, and few-shot formatters are in `ragu/triplet/prompts.py`. Listings show system prompts; user messages supply the chunk text (Stage 2 adds the entity list).

Stage 1: entity extraction (system prompt, abridged).

You are an expert entity extraction system. Identify and extract all

```
meaningful entities from the provided text, returning,
for each, an
**entity_name** (normalized canonical name), an
**entity_type**, and a
**description** (context-rich). [...]

**RULES** [...]
- **No hallucination**: only entities explicitly
  mentioned or clearly
  implied by the text.
- Use only these entity types: {{ entity_types }}.
```

Return valid JSON matching the schema, in: {{ language }}.

Stage 2: relation extraction (system prompt, abridged).

You are an expert relation extraction system. Given a source text and the entities already extracted from it, return for each relationship a

```
**source_entity** and **target_entity** (each MUST
exactly match a
provided entity name), a **relation_type**, a
**description**, and a
**relationship_strength** (integer 1--5). [...]
```

RULES [...]

```
- **source_entity**/**target_entity** must exactly
  match provided names.
- Use only these relation types: {{ relation_types }}.
```

Return valid JSON matching the schema, in: {{ language }}.

Verbose generation prompt (RAGU default).

Answer the query by summarizing relevant information from the context and, if necessary, well-known facts. If you do not know the correct answer, state that; do not include unsupported information.

Query: {{ query }}

Context: {{ context }}

Answer in: {{ language }}.

Terse generation prompt.

Answer the query as briefly as possible using the context. Output ONLY the direct answer (a single entity, name, number, date, or a short semicolon-separated list); no explanations, full sentences, restating, markdown, or citations. If unsupported by context, output: I don’t know.

Query: {{ query }}

Context: {{ context }}

Answer in: {{ language }}.