HEMLOCK - HYBRID ENTITY MULTI-HOP LINKAGE OVER CO-OCCURRENCE KNOWLEDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

A multi-hop question needs documents it reaches only through entities it never names, and on 19–51% of questions ColBERTv2's top 30 misses one. Closing that gap usually costs a generative call per indexed passage or reasoning step. HEMLOCK makes none: a bipartite graph links text segments to the entities they mention, and a Personalised PageRank walk seeded from ColBERTv2 reaches bridge segments through shared entities for a cross-encoder to rerank. It spends zero LLM tokens at index and query time, serves a query in 96 ms and needs no per-corpus tuning. On HotpotQA, 2WikiMultihopQA and MuSiQue the walk adds 4.2–5.4 pp of passage recall@10 over the same pipeline without it; no other system we evaluate that makes no generative calls retrieves more, LinearRAG at its published configuration included. HEMLOCK reaches 85–95% of the document recall@10 of HippoRAG 2, whose graph is LLM-built, and exceeds IRCoT's by 14.0–16.3 pp. Where ColBERTv2's top-30 misses a gold document, no reranker over that list can recover it; the walk lifts 49.5, 39.9 and 21.7% of them into the top ten, raising exact match under a frozen reader.

1 INTRODUCTION

Multi-hop question answering requires chaining information across documents to reach an answer no single document contains. In "In what year was the director of *Inception* born?", the first hop is easy because the query names *Inception*; the second is hard, because the document about Nolan's birth year shares neither lexical nor semantic overlap with the query and connects only through a bridge entity the retriever has to discover for itself. The problem is widespread: even ColBERTv2, a strong late-interaction retriever, misses a gold document in its top 30 on 19–51% of questions across HotpotQA (Yang et al., 2018), 2WikiMultihopQA (Ho et al., 2020) and MuSiQue (Trivedi et al., 2022) (Section 5.2).

Lexical retrieval (Robertson & Zaragoza, 2009), dense retrieval (Karpukhin et al., 2020), their reciprocal-rank-fusion hybrid and token-level late interaction (Santhanam et al., 2022) all score a passage against the query alone, so a bridge passage the query does not name is indistinguishable from any other passage it does not name. The systems that do bridge add a generative language model: HippoRAG 2 (Gutiérrez et al., 2025) extracts triples with an LLM at every indexed passage and IRCoT (Trivedi et al., 2023) issues a chain-of-thought call at every reasoning step, both at substantial deployment cost.

We propose HEMLOCK, which exploits structure already present in text. At index time, extractive components identify named entities and link each text segment to the canonical entities it mentions; at query time, a Personalised PageRank (PPR) walk seeded from a ColBERTv2 hit-list reaches passages connected through unnamed bridge entities. No generative language model is involved at either stage, no model larger than 209M parameters is used at either, none larger than 110M runs at query time, and the graph walk itself uses no model at all. Where ColBERTv2 fails, the walk reaches two thirds of the gold documents HEMLOCK still misses without promoting them into the top ten (Section 5.5).

Contributions.

1. **Multi-hop retrieval at $0 per query.** HEMLOCK composes coreference resolution, named entity recognition, entity canonicalisation, a first-stage retriever and PPR into one extractive

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.

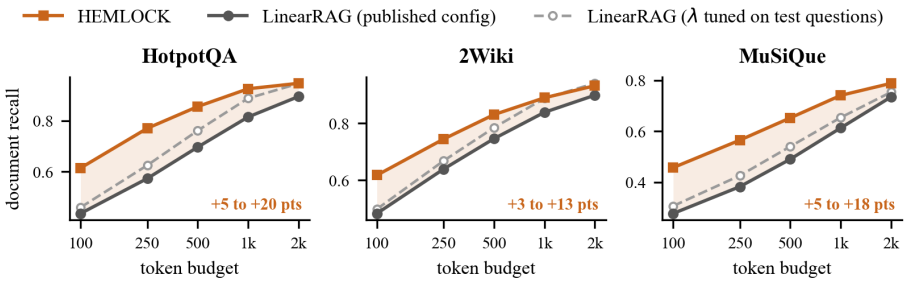


Figure 1: **HEMLOCK retrieves more gold evidence per input token than LinearRAG.** Document recall against retrieved-token budget, both systems at one fixed configuration with lists depth-matched at 30. Solid: LinearRAG at its published hyperparameters; the shaded gap and in-panel ranges are measured against it. Dashed: LinearRAG with its dense-mixing weight λ tuned on the test questions, against which HEMLOCK's lead narrows but holds through 500 tokens. The advantage carries to answers: under a frozen reader, LinearRAG is never significantly ahead on exact match while HEMLOCK reads 26–36% fewer tokens (Section 5.4).

pipeline: no generative model at either stage, nothing larger than 209M parameters at index time or 110M at query time, and 96 ms per query. It reaches 85–95% of the recall@10 of a graph retriever whose index costs thousands of generative calls.

2. **The graph reaches evidence the first-stage retriever never ranks.** A reranker can only reorder the list it is given, and on 19 to 51% of questions ColBERTv2's top-30 misses a gold document altogether. Walking the entity graph from those seeds recovers 49.5, 39.9, and 21.7% of them and raises exact match on those queries under a frozen reader (Section 5.2).
3. **More gold evidence per token of context.** Context is billed per token. HEMLOCK retrieves 3-sentence segments rather than whole passages and returns more gold evidence than LinearRAG at every budget from 100 to 2,000 tokens on all three benchmarks (Figure 1). Re-indexing LinearRAG on the same segments narrows the margin without closing it.

2 Related Work

Retrieval without graph structure. BM25 (Robertson & Zaragoza, 2009) ranks by lexical overlap and Dense Passage Retrieval (Karpukhin et al., 2020) by semantic similarity, with reciprocal rank fusion combining the two; ColBERTv2 (Santhanam et al., 2022) replaces single-vector retrieval with token-level late interaction via MaxSim aggregation, and is the strongest non-LLM multi-hop baseline available off the shelf. Multi-hop Dense Retrieval (Xiong et al., 2020) and Think-on-Graph 2.0 (Ma et al., 2025) extend retrieval with recursive reformulation and KG navigation respectively.

Graph retrieval without a language model. The closest published system we can compare against is LinearRAG (Zhuang et al., 2026), which builds a passage-entity graph with lightweight entity extraction and no relation extraction at index time, then retrieves by activating query-relevant entities and aggregating their importance back onto passages. Its graph adds a passage-passage adjacency chain to the bipartite edges, its restart mass sits on entity nodes rather than passages, and it mixes a dense channel under a per-corpus weight λ, where HEMLOCK's graph layer reads no embeddings. Section 5.4 and Appendix K compare the two.

Related entity-graph designs. SPRIG (Wang, 2025) is closest in design: an NER-built entity–document bipartite graph, PPR seeded from BM25 and dense hits, and entities discounted by document frequency. HEMLOCK seeds from ColBERTv2 segments, cuts the pool by PPR mass before reranking, and leaves edges unweighted, taking the same discount from the closed form of Section 3.3. Its corpora are larger, so the published numbers are not comparable. EHRAG (Song et al., 2026) adds entity embeddings to a co-occurrence hypergraph, and MixPR (Alonso & Millidge, 2024) mixes sparse PageRank variants in place of long-context inference.

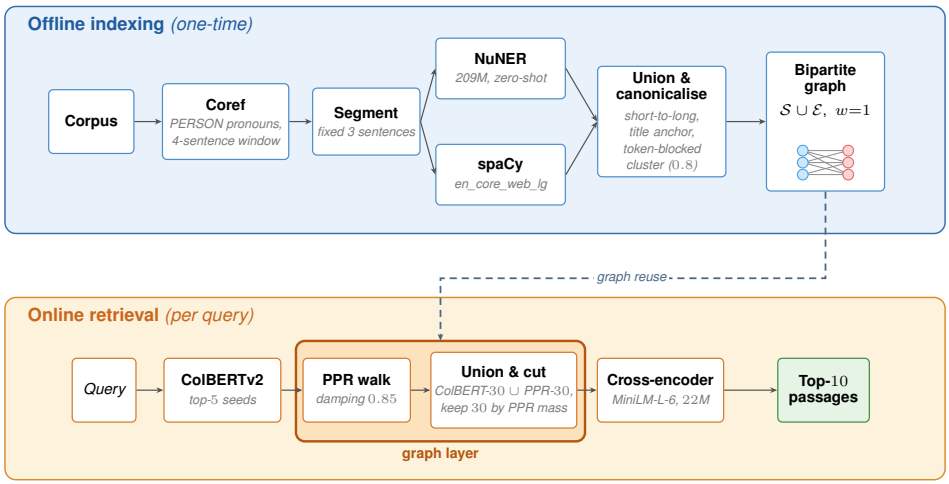


Figure 2: **End-to-end HEMLOCK pipeline. Offline (left):** coreference resolution, uniform 3-sentence segmentation, NER and entity canonicalisation produce a bipartite segment-entity graph. **Online (right):** ColBERTv2's top-k hits seed PPR over the bipartite graph, and the resulting candidates are reranked by a cross-encoder. All components are extractive.

Bipartite PageRank and link prediction. PPR on a bipartite graph collapses onto a projected segment-to-segment operator (Deng et al., 2009; Yang, 2022; Liu & Luo, 2024). Its leading term needs no iteration and scores a pair of nodes by their shared neighbours, each discounted by its degree, which is the resource-allocation index of link prediction (Zhou et al., 2009). Section 3.3 applies this identity to the segment–entity graph and measures what it implies for retrieval.

Ranking as the bottleneck. Iterative GraphRAG retrieval helps multi-hop questions by promoting bridge documents into the leading ranks (Guo et al., 2025), and for conversational memory a deterministic pipeline of NER-weighted matching and reranking outperforms graph structure (Derehag et al., 2026). Section 5.5 measures the same bottleneck in an LLM-free graph retriever.

Graph retrieval with a language model. IRCoT (Trivedi et al., 2023) alternates retrieval and chain-of-thought reasoning, paying one LLM call per reasoning step. Among graph-augmented approaches (De Cao et al., 2019; Yasunaga et al., 2021; Edge et al., 2024; Mavromatis & Karypis, 2024), the most relevant to our work is HippoRAG (Gutiérrez et al., 2024), which showed that PPR over LLM-extracted knowledge graphs matches iterative retrieval in a single step, using an LLM to extract typed, directional triples; the baseline we run is its successor HippoRAG 2 (Gutiérrez et al., 2025), which adds a per-query LLM filter over candidate triples before the walk. HEMLOCK replaces that LLM extraction with extractive NER and undirected entity co-occurrence edges, trading relation-type information for zero LLM calls at index time.

3 Method

HEMLOCK runs in two phases: offline indexing builds a graph index from the corpus, and online retrieval traverses it (Figure 2). ColBERTv2 supplies seed segments, PPR walks the bipartite segment-entity graph from those seeds to surface bridge passages the seeds do not, and a cross-encoder reranks the resulting pool; all three are extractive.

3.1 Indexing

Coreference resolution. Before segmentation, a conservative rule-based resolver replaces third-person singular PERSON pronouns with the most recent PERSON entity within four sentences, so a segment reading "she won the Academy Award" keeps its edge to Emma Stone.

Segmentation. We segment documents into uniform 3-sentence chunks; boundaries are fixed by sentence count alone and depend on no model.

Entity extraction and canonicalisation. We extract named entities using the union of NuNER (Bogdanov et al., 2024) (209M, zero-shot) and spaCy en_core_web_lg (Honnibal et al., 2020), with span-level deduplication of overlapping extractions. Canonicalisation proceeds in three stages (Appendix A): within-document short-to-long resolution maps abbreviations to longest forms; mentions are fuzzy-matched against document titles (similarity threshold 0.8); and global token-blocked fuzzy clustering handles remaining surface-form variation.

Graph construction. We construct a bipartite graph $G = (S \cup E, W)$ where S is the set of segments, E is the set of canonical entities, and every segment is joined to each canonical entity it mentions by an edge of unit weight:

$$w(s, e) = 1. \tag{1}$$

No learned, embedded or corpus-fitted quantity enters the graph. A weighted variant mixes term frequency with entity-segment cosine similarity from a 1.5B-parameter embedder,

$$w_\alpha(s, e) = \alpha \, \mathrm{tf_norm}(e, s) + (1 - \alpha) \max(\cos(v_e, v_s), 0), \tag{2}$$

and ties or loses to binary edges (Table 6). HEMLOCK uses binary edges.

3.2 Retrieval

HEMLOCK retrieves the top-k_0 seed segments (k_0=5) from a ColBERTv2 (Santhanam et al., 2022) late-interaction index over the same 3-sentence segments, which gives lexically-grounded entry points at the cost of a one-time ~ 25 s PLAID indexing pass. The walk is seeded from those segments rather than from the query's own entities, which removes query-side entity extraction. We run PPR from them over the bipartite segment-entity graph with damping 0.85, the canonical PageRank value (Brin & Page, 1998), against a precomputed sparse column-stochastic transition matrix. Appendix B traces the walk on a single query.

Candidate assembly. The candidate pool is the union of ColBERTv2's top-d segments and the top-d segments by PPR mass, with d=30. The union is ranked by PPR score alone, with no query-similarity term (the `raw` in `ppr_colbert_raw_ce`; Appendix G ablates the alternative), and truncated back to d. The cross-encoder `ms-marco-MiniLM-L-6-v2` (Nogueira & Cho, 2020) (22M) then scores those d pairs and returns the top ten. A segment that shares rare entities with several seeds outranks one that matches the query but shares no entity with the seeds: this is how a bridge segment the query never names reaches the reranker, and how a segment ColBERTv2 ranked highly can be dropped before it. Neither d nor k_0 was tuned; Appendix N varies d.

3.3 The graph stage in two sparse passes

PPR from segment seeds collapses onto a segment-to-segment operator, the hidden transition matrix of Deng et al. (2009): the walk alternates sides, so one step from a seed lands *entirely* on entities and every odd-order term vanishes on segments. Its leading term is two sparse passes over the incidence lists (Yang, 2022; Liu & Luo, 2024):

$$a(e) = \sum_{s \in \mathcal{S}_0} r_s \frac{w(s, e)}{\deg_S(s)}, \qquad \mathrm{CF}(s') = \sum_e a(e) \frac{w(s', e)}{\deg_E(e)}. \tag{3}$$

Both passes apply one rule: a node divides what it holds evenly among its neighbours. Each seed carries mass $r_s = 1/k_0$ and splits it over the $\deg_S(s)$ entities it mentions; every entity then splits its accumulated $a(e)$ over the $\deg_E(e)$ segments that mention it, so a rare entity passes a large share to each and a common one passes little. On binary edges CF from a single seed is Equation 11 of Zhou et al. (2009), and a rare bridge therefore outranks a generic co-mention. Ranking by CF matches the walk's recall@10 to within 0.1 pp on all three benchmarks. Removing the $\deg_E$ division costs 4.3–5.1 pp (Table 6). Scoring falls from 32–59 ms per query to 1.3–3.2 ms, a stage-level figure we do not extend to end-to-end latency. Appendix J gives the expansion, identity checks, training-split figures and scoring cost.

4 EXPERIMENTAL SETUP

Datasets. HotpotQA (Yang et al., 2018): 1,000 dev questions from the bridge subset in the distractor setting, whose context paragraphs are concatenated by title into documents. 2WikiMultihopQA (Ho et al., 2020): 999 2-hop compositional, comparison and inference questions. MuSiQue (Trivedi et al., 2022): the first 1,000 2-hop dev questions. All retrievers search one shared corpus per dataset, thousands of documents rather than the $\sim$10 context paragraphs supplied per question. Table 1 fixes the evaluation protocol.

Reading the numbers. Unless a caption says otherwise, every number in the main text is HotpotQA dev, the binary graph of Equation 1, and passage recall. Number triples run HotpotQA, 2WikiMultihopQA, MuSiQue in that order. The exception: HotpotQA train wherever HippoRAG 2 or IRCoT appear. Appendix D maps every HEMLOCK figure to its split, graph and granularity.[1]

Table 1: **Evaluation protocol.** The corpus is shared across all retrievers within a dataset. Latency is reported on the GPU listed.

Dataset	Split	Questions	Documents	Segments	GPU
HotpotQA	dev	1,000	9,836	17,131	H200
2WikiMultihopQA	—	999	6,193	10,103	B200
MuSiQue	dev	1,000	12,493	18,690	B200

Baselines. We compare against four lexical and dense configurations (BM25, Dense, and a reciprocal-rank-fusion hybrid, each optionally reranked by the same `ms-marco-MiniLM-L-6-v2` cross-encoder HEMLOCK uses), ColBERTv2 (Santhanam et al., 2022) in its native indexing mode, and the three graph or LLM-augmented systems of Section 2: LinearRAG (Zhuang et al., 2026), IRCoT (Trivedi et al., 2023) and HippoRAG 2 (Gutiérrez et al., 2025). HippoRAG 2 keeps its released 7B embedder, NV-Embed-v2 (Lee et al., 2024), and uses `gpt-4o-mini` for its LLM calls. **RRF+CE** is the strongest production-style baseline using neither late interaction nor graph structure, and **ColBERTv2+CE**, HEMLOCK without the walk, is the comparator for every claim about the graph. Checkpoints, hyperparameters, API pricing and per-system configuration are in Appendix C.

Metrics. Recall@k is the fraction of a question's gold documents whose titles appear in what was retrieved, averaged over questions. The prefix names the budget: passage recall@10 allows ten segments and document recall@10 allows ten documents, with HEMLOCK's segments resolved to documents first, so document recall runs higher at the same k. Table 3 and Section 5.4 use document recall, because HippoRAG 2 and LinearRAG retrieve documents. Differences are given in percentage points (pp), where 1 pp is 0.01 recall or exact match. Significance is a paired bootstrap (n=10,000) over identical query indices, at $p < 0.001$ unless a value is given.

Reader. End-to-end answers come from one frozen reader, `qwen3-8b` (Yang et al., 2025) at temperature 0, with prompt and decoding identical across systems so that only the retrieved context differs; we report SQuAD exact match, bounded below by a closed-book run and above by supplying the gold context. Hardware, API and reader details are in Appendix C.

5 RESULTS

5.1 HEADLINE RESULTS

HEMLOCK is the strongest retriever we evaluate that makes no generative calls. Holding the seed retriever and reranker fixed, the graph walk adds +4.2, +5.4, and +4.5 pp of recall@10 over ColBERTv2+CE (paired bootstrap, all $p < 10^{-4}$). Against the strongest baseline on each benchmark the margin is +3.8 pp over RRF+CE on HotpotQA and +5.2 over native ColBERTv2 on 2WikiMultihopQA. On MuSiQue the strongest baseline is ColBERTv2+CE itself. Holding

[1] No component of HEMLOCK is trained or fitted. Every model is off the shelf and the damping factor, k_0 and d are fixed a priori.

Table 2: **LLM-free retrieval: passage recall@10.** Every system here spends zero LLM tokens at index and query time. All run on the same coref-resolved 3-sentence-chunked corpus except ColBERTv2†, indexed natively at ColBERT's default 256-token passages. n=1000 / 999 / 1000.

Retriever	HotpotQA	2Wiki	MuSiQue
Dense	0.6845	0.6216	0.4840
BM25	0.7715	0.7397	0.5000
BM25+CE	0.8485	0.7843	0.5835
RRF (BM25+Dense)+CE	0.8740	0.8163	0.6400
ColBERTv2†	0.8490	0.8233	0.6490
ColBERTv2+CE	0.8705	0.8213	0.6635
HEMLOCK (`ppr_colbert_raw_ce`)	**0.9120**	**0.8754**	**0.7085**

segmentation fixed as well, by running ColBERTv2 over the same 3-sentence segments, leaves the walk's contribution intact (Appendix F).

The margins track how much of its own evidence each benchmark's questions name. The graph's contribution is concentrated on questions whose text names exactly one of their two gold documents (+6.2, +11.4, +8.4 pp); it is not significant where the question names neither, and negative where it names both, since ColBERTv2+CE already scores 0.95–0.98 there (Appendix O).

LLM-based retrievers as bounds. The two LLM-based retrievers are not cost-comparable with the systems above and we report them as bounds rather than as baselines: HippoRAG 2 spends one generative call on every indexed passage, and IRCoT one on every reasoning step. Table 3 places HEMLOCK against both at matched split.

Table 3: **What a generative language model buys, and what it costs.** Document recall@10: every row is scored at document granularity, since HippoRAG 2 retrieves documents. HippoRAG 2's indexing calls are one-time; it also makes one filter call per query. HotpotQA is train for every row, so the comparison is split-matched; on dev HEMLOCK reads 0.9235. Dollar costs are in Table 7.

Retriever	Index calls	Calls/query	HotpotQA	2Wiki	MuSiQue
`IRCoT` (qwen3-8b, T=8)	0	1.5–2.3	0.7625	0.7252	0.5885
HippoRAG 2 (LLM-OpenIE)	6,193–21,100	1	**0.9890**	**0.9400**	**0.8560**
HEMLOCK	**0**	**0**	0.9220	0.8879	0.7285

HEMLOCK reaches 85 to 95% of HippoRAG 2's recall@10 with no generative model anywhere in the pipeline, and beats IRCoT by 16.0, 16.3, and 14.0 pp. HippoRAG 2 is the ceiling this design gives up: LLM-extracted typed triples retrieve more gold than extractive co-occurrence does, on all three benchmarks.

5.2 Recovery where the seed retriever fails

Graph traversal matters on the queries the seed retriever cannot serve, and those are identifiable without reference to any other system: the queries where ColBERTv2's top-30 misses at least one gold document (n=191, 271, and 510). ColBERTv2+CE only reranks that top-30, so its recall on the 192, 273, and 535 missing documents is zero by construction, verified on every one. HEMLOCK recovers 49.5, 39.9, and 21.7% of them into its top ten (Figure 3), raising recall@10 on those queries by +24.4, +19.6, and +9.9 pp and exact match by +20.9, +10.7, and +7.7 (HotpotQA dev).

ΔEM uses the frozen reader at k=10; on HotpotQA train the picture is the same (213 queries, 50.0% recovered, +19.7 pp EM). Recovery falls as the regime widens: ColBERTv2 misses a gold document on 19% of HotpotQA queries and 51% of MuSiQue queries, and the recovered fraction falls from 49.5% to 21.7%.

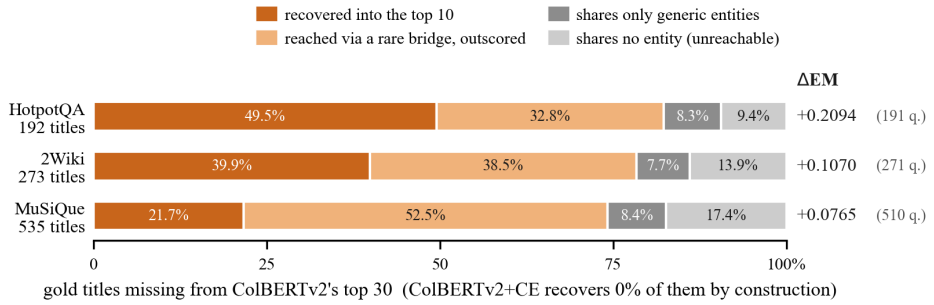


Figure 3: **What happens to the evidence ColBERTv2 misses.** Gold documents absent from ColBERTv2's top 30, split by what HEMLOCK does with them. ColBERTv2+CE only reranks that top 30, so it recovers none of them. ΔEM is HEMLOCK minus ColBERTv2+CE on the same queries, frozen reader at k=10.

Table 4: **Exact match under the frozen reader**, at k=10 and k=20. Prompt, decoding and reader are identical across rows; only the retrieved passages differ. IRCoT and HippoRAG 2 call an LLM and were not run on MuSiQue or beyond k=10. HotpotQA is train. Closed-book and gold-context bounds, and $k \in \{3, 5\}$, are in Appendix P.

	HotpotQA		2Wiki		MuSiQue	
System	k=10	k=20	k=10	k=20	k=10	k=20
RRF (BM25+Dense)+CE	0.5710	0.5770	0.2392	0.2342	0.1190	0.1200
ColBERTv2+CE	0.5440	0.5690	0.2352	0.2312	0.1220	0.1310
HEMLOCK	0.6220	**0.6500**	0.2793	**0.2933**	**0.1740**	**0.2080**
IRCoT (frozen reader)	0.5490	—	0.2052	—	—	—
HippoRAG 2	**0.6700**	—	**0.2893**	—	—	—

5.3 End-to-end question answering

Recall@10 is a proxy. We test whether the retrieval differences survive to answers by supplying each system's ranking to the frozen reader of Section 4, varying only the number of passages (Table 4; Table 21 in Appendix P gives every value).

Against RRF+CE (Appendix P) at k=10, HEMLOCK gains +5.1, +4.0, and +5.5 pp, and the advantage grows with k: at k=20 it reaches +7.3 on HotpotQA and +8.8 on MuSiQue. Under the same reader HEMLOCK leads IRCoT by +7.3 and +7.4 on HotpotQA and 2WikiMultihopQA; IRCoT's own chain-of-thought reader, which spends further LLM calls, reaches 0.5750, 0.4054, and 0.2480 at k=10, above HEMLOCK on 2WikiMultihopQA and MuSiQue. HippoRAG 2 answers more accurately than HEMLOCK at both depths where it can be measured, which is what its LLM-built graph buys (Table 3), although it loses accuracy from k=5 to k=10 where HEMLOCK gains on every benchmark. On HotpotQA dev the ordering replicates, with HEMLOCK +3.8 over RRF+CE at k=10 (Appendix P).

5.4 Comparison with LinearRAG

LinearRAG's released passages are far longer than our segments (on 2WikiMultihopQA, ∼4,400 characters against 267; Appendix K), so recall here is document-level.

At their published hyperparameters HEMLOCK retrieves more gold on all three benchmarks (Table 5). Tuning λ on the test questions is worth 4.2–6.6 pp to LinearRAG, and against that oracle HEMLOCK still leads on MuSiQue, is level on HotpotQA and trails on 2WikiMultihopQA. HEMLOCK has no per-corpus hyperparameter; choosing the best of fifteen variants per dataset would gain it at most 0.6 (Appendix L). LinearRAG reports gains over HippoRAG 2, while HEMLOCK leads LinearRAG and trails HippoRAG 2 (Table 3). The orderings do not chain: LinearRAG scores containment

Table 5: **Document recall@10 against LinearRAG.** *Published*: the per-dataset hyperparameters LinearRAG ships. λ *oracle*: its dense-mixing weight λ swept on our test questions, best value kept (Appendix K). Paired bootstrap n_{iter}=10,000 on identical query indices. Against the oracle, p=0.123 on HotpotQA and p=0.0003 on MuSiQue.

	LinearRAG			vs published			vs oracle
Benchmark	published	λ oracle	HEMLOCK	Δ	95% CI	p	Δ
HotpotQA	0.8495	0.9150	**0.9235**	+0.0740	[+0.056, +0.092]	$< 10^{-4}$	+0.0085
2Wiki	0.8624	**0.9039**	0.8879	+0.0255	[+0.006, +0.046]	0.007	−0.0161
MuSiQue	0.6430	0.6915	**0.7285**	+0.0855	[+0.062, +0.109]	$< 10^{-4}$	+0.0370

accuracy rather than recall, and its comparison runs every system, HippoRAG 2 included, with all-mpnet-base-v2 in place of HippoRAG 2's NV-Embed-v2 (Appendix K).

Under the frozen reader of Section 5.3, LinearRAG's rankings never yield significantly higher exact match than HEMLOCK's at any $k \in \{5, 10, 20\}$; HEMLOCK leads significantly on HotpotQA dev at k=5 and 10 (+5.1, +4.6 pp) and on MuSiQue at k=20 (+4.4; all $p \leq 3 \times 10^{-4}$, within a Bonferroni correction for nine tests), while supplying the reader 26–36% fewer input tokens (Table 16). We also compare recall at matched input-token budgets. At LinearRAG's published configuration HEMLOCK retrieves more gold per input token at every budget from 100 to 2,000 tokens on all three benchmarks (Figure 1), and through 500 tokens with λ re-tuned (Appendix K). Part of this is unit size: our units run 82–86 tokens against their 116–145. Re-indexing LinearRAG on our segments narrows the margin but does not close it (Appendix K).

On HotpotQA dev and the same H200, HEMLOCK serves at 96 ms per query against LinearRAG's 149 ms at the document granularity of Table 5, while LinearRAG indexes our corpora faster, 40–87 s against ∼15 min.

LinearRAG's walk reduces to a score of the same form as Section 3.3, with the discount entering as $1/\deg_E$ rather than $1/\deg_E^2$ (Appendix K), so the margin does not come from the scoring rule. Section 5.5 attributes most of the graph's contribution to the PPR-ordered cut rather than to the extra candidates, and the two designs differ in that cut and in their seeds.

5.5 The remaining bottleneck is ranking

Candidates and ordering. The graph contributes both extra candidates and the PPR ordering that decides which 30 of the union reach the cross-encoder. Rescoring the cached pools separates the two: ranking the whole union is worth +1.7, +1.1, and +0.3 pp, and cutting it by PPR score adds +2.5, +4.4, and +4.2. Every gold segment HEMLOCK loses relative to ColBERTv2+CE was removed by that cut before the cross-encoder scored it (Appendix M).

Where HEMLOCK still misses gold, the evidence has usually been retrieved and scored already: recall@30 over the reranker's pool is 0.9480, 0.9404, and 0.8000, against recall@10 of 0.9120, 0.8754, and 0.7085, a ranking loss that exceeds the walk's own contribution on 2WikiMultihopQA and MuSiQue. Of the titles ColBERTv2 misses that HEMLOCK does not recover, about two thirds share a rare bridge entity with a seed (Figure 3).

Promotion of graph-only segments. Gold segments ColBERTv2 had already retrieved reach the top ten 96–98% of the time; gold segments only the graph reached manage 59.3, 42.8, and 28.7%. A reranker trained on different data, `bge-reranker-base` (Xiao et al., 2024) (278M), lifts that to 87.6, 59.7, and 40.2% and widens the graph's seed-matched contribution by +0.5, +1.4, and +2.2 pp on identical pools (Appendix N).

5.6 Cost

HEMLOCK makes no LLM calls at index or query time and serves a query in 96 ms, while HippoRAG 2 makes 6,193–21,100 one-time OpenIE calls ($1.24–$4.22 per benchmark) plus one per query, and IRCoT 1.5–2.3 per query (Table 7, Appendix C). Against ColBERTv2+CE, HEMLOCK

adds 71.5 ms per query for $+4.2$ to $+5.4$ pp of recall@10, and it is roughly 4.5–$7\times$ faster than IRCoT served locally, a ratio we report as indicative because the two ran on different GPUs. Indexing scales linearly over four HotpotQA dev corpora of up to 6.84M tokens (exponent 1.045, $R^2{=}0.995$; the graph layer alone 1.002).

6 Ablations

Table 6: **Component ablation, passage recall@10.** One component varied at a time against HEMLOCK as specified in Section 3. Candidate-cut and reranker controls: Appendices M and N.

Component	Variant	HotpotQA	2Wiki	MuSiQue
—	**HEMLOCK** (binary edges)	**0.9120**	**0.8754**	**0.7085**
Graph	removed (ColBERTv2+CE)	0.8705	0.8213	0.6635
Edges	weighted, Eq. 2 at $\alpha{=}0.5$	0.9045	0.8739	0.7075
Edges	term frequency only ($\alpha{=}1$)	0.8785	0.8564	0.6945
Edges	cosine only ($\alpha{=}0$)	0.9105	0.8744	0.7115
Walk	two passes (Eq. 3)	0.9110	0.8754	0.7085
Walk	resource-allocation score only	0.9085	0.8769	0.7040
Walk	no $1/\deg_E$ discount	0.8580	0.8343	0.6560

Removing the $1/\deg_E$ discount costs more than removing the graph on HotpotQA and MuSiQue, where the walk without it scores below no graph at all (0.8580 against 0.8705, 0.6560 against 0.6635). The weighted and cosine-only edge variants land within 0.8 pp of binary edges, while term frequency alone falls 1.4 to 3.4 pp below. Binary edges lose nothing, and the graph needs no embedding model.

Other design choices. Uniform 3-sentence chunks match or beat an adaptive segmentation (Appendix E), PPR beats depth-bounded BFS_2 with ColBERTv2 seeds (Appendix H), and Appendix G ablates the composite scorer.

7 Limitations

Our evaluation uses $1{,}000$-question subsets for HotpotQA and MuSiQue and 999 questions for 2WikiMultihopQA, on evaluation corpora of 0.68M to 1.33M tokens (only the indexing-scaling runs of Section 5.6 go larger, to 6.84M), and MuSiQue's 3- and 4-hop chains are untested.

The graph also needs the query to name some of its own evidence: on questions naming none of their gold documents it contributes $+0.4$ pp on MuSiQue and $+1.8$ on HotpotQA, neither significant. HEMLOCK improves multi-hop retrieval that already has a lexical foothold; it does not supply one. Some residual misses may lie outside entity-graph retrieval altogether: where HEMLOCK and LinearRAG both fail, they miss the same documents far more often than chance (Appendix Q). Token-blocked canonicalisation both conflates similar surface forms and misses synonyms sharing no token.

End-to-end results come from one frozen reader whose gold-context ceiling is 0.776, 0.469, and 0.440. On HotpotQA dev that ceiling falls to 0.594, although its gold context contains the answer string about as often as train's, so absolute exact match reflects the reader rather than the retrieval.

8 Conclusion

We presented HEMLOCK, a multi-hop retriever that makes no generative calls at index or query time. Among the systems we evaluate that make no such calls, with LinearRAG at its published configuration, it has the highest recall@10 on all three benchmarks. It reaches 85–95% of the recall of HippoRAG 2 and exceeds the recall of IRCoT. The walk reduces to one sparse pass, and the degree discount in it carries the result. Where ColBERTv2 misses a gold document, the graph recovers 49.5%, 39.9%, and 21.7% of them and raises exact match on those questions; where it still fails, the reranker leaves two thirds below the top ten. Chains deeper than two hops remain untested.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods and to support the interpretation of results: LLM-based coding assistants wrote and refactored parts of the experimental and analysis code, and writing assistants drafted prose describing results whose numbers we had already computed. We have not used generative AI tools to propose or refine the hypotheses, to design or give feedback on the experimental methodology, to develop the conceptual framing, to formulate mathematical claims, to supply critical ingredients for or to write proofs, or to clean and reformat the datasets, which are used at their public releases. Generating synthetic data, translation, and qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the paper for readability, to write and revise the plotting code behind the figures, to search for and identify related literature, and to format references.

We have reviewed all AI-assisted work. Every number in the paper comes from a stored result file written by an experimental run and was checked against that file; no reported quantity was taken from a model's output in place of a measurement. AI-assisted code was read and executed by the authors, and its outputs were checked against independently computed values wherever such a check was available (Appendix J). Every reference a tool suggested was located and read before being cited. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Generative models also appear inside the experiments, as evaluated baselines and as the frozen reader. That use is part of the object of study and is described in Section 4 and Appendix C.

ETHICS STATEMENT

This work evaluates retrieval systems on public question-answering benchmarks, used as released and without modification. It involves no human subjects, no new data collection, and no personally identifying information beyond what those public corpora already contain. HEMLOCK itself is extractive and makes no generative calls, so it adds no capacity to produce fluent unsupported text; the end-to-end results pass retrieved context to a hosted reader and report exact match against the benchmark answers.

REPRODUCIBILITY STATEMENT

Section 4 and Table 1 fix the evaluation protocol: the datasets, the split, filter and ordering behind each question subset, the shared per-dataset corpus, and the hardware on which each measurement ran. Section 3 specifies the pipeline declaratively, from the coreference rule and the 3-sentence segmentation through the NER backends, the canonicalisation stages, the binary edge definition of Equation 1, the seed count k_0, the damping factor, the candidate depth d, and the cross-encoder checkpoint. Appendix A gives the canonicalisation thresholds and the stored artefacts. No component is trained or fitted, and k_0, d and the damping factor are fixed a priori rather than swept per corpus, so there is no training run or hyperparameter search to reproduce. Appendix C lists each baseline's checkpoint, released implementation and configuration, the frozen reader's decoding settings, and the hardware and API pricing behind every cost figure. Appendix D maps each HEMLOCK recall figure to its split, graph variant and granularity, and Appendix J reports the identity checks behind the two-pass reduction with their numerical tolerances. Significance tests are paired bootstraps over identical query indices at the iteration count given in Section 4. Code is not released with this submission; every model in the pipeline is a public off-the-shelf checkpoint, and the specification above is intended to be sufficient for reimplementation.

REFERENCES

Nicholas Alonso and Beren Millidge. Mixture-of-PageRanks: Replacing long-context with real-time, sparse GraphRAG. *arXiv preprint arXiv:2412.06078*, 2024.

Sergei Bogdanov, Alexandre Constantin, Timothée Bernard, Benoit Crabbé, and Etienne P Bernard. NuNER: Entity recognition encoder pre-training via LLM-annotated data. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 11829–11841, Miami, Florida, USA, November

2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.660. URL https://aclanthology.org/2024.emnlp-main.660/.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117, 1998. URL https://snap.stanford.edu/class/cs224w-readings/Brin98Anatomy.pdf.

Nicola De Cao, Wilker Aziz, and Ivan Titov. Question answering by reasoning across documents with graph convolutional networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2306–2317, 2019.

Hongbo Deng, Michael Lyu, and Irwin King. A generalized co-hits algorithm and its application to bipartite graphs. pp. 239–248, 06 2009. doi: 10.1145/1557019.1557051.

Jesper Derehag, Carlos Calva, and Timmy Ghiurau. SmartSearch: How ranking beats structure for conversational memory retrieval. *arXiv preprint arXiv:2603.15599*, 2026.

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.

Kai Guo, Xinnan Dai, Shenglai Zeng, Harry Shomer, Haoyu Han, Yu Wang, and Jiliang Tang. Beyond static retrieval: Opportunities and pitfalls of iterative retrieval in graphrag, 2025. URL https://arxiv.org/abs/2509.25530.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Nuria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. Zenodo, 2020.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024. URL https://arxiv.org/abs/2405.17428.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Haoyu Liu and Siqiang Luo. Bird: Efficient approximation of bidirectional hidden personalized pagerank. *Proceedings of the VLDB Endowment*, 17(9):2255–2268, 2024.

Shengjie Ma, Chengjin Xu, Xuhui Jiang, Muzhi Li, Huaren Qu, Cehao Yang, Jiaxin Mao, and Jian Guo. Think-on-graph 2.0: Deep and faithful large language model reasoning with knowledge-guided retrieval augmented generation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 52782–52806, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/830b1abc6d2da85f23d41169fa44d185-Paper-Conference.pdf.

Costas Mavromatis and George Karypis. Gnn-rag: Graph neural retrieval for large language model reasoning. *arXiv preprint arXiv:2405.20139*, 2024.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020. URL https://arxiv.org/abs/1901.04085.

OpenAI. GPT-4o mini: Advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/, July 2024. Released July 18, 2024.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009. doi: 10.1561/1500000019.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3715–3734, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.272. URL https://aclanthology.org/2022.naacl-main.272/.

Yifan Song, Xingjian Tao, Zhicheng Yang, Yihong Luo, and Jing Tang. EHRAG: Bridging semantic gaps in lightweight GraphRAG via hybrid hypergraph construction and retrieval. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 24638–24652, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1233. URL https://aclanthology.org/2026.findings-acl.1233/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022. doi: 10.1162/tacl_a_00475. URL https://aclanthology.org/2022.tacl-1.31/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.557. URL https://aclanthology.org/2023.acl-long.557/.

Qizhi Wang. Democratizing GraphRAG: Linear, CPU-only graph retrieval for multi-hop QA. *arXiv preprint arXiv:2602.23372*, 2025.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, et al. Answering complex open-domain questions with multi-hop dense retrieval. *arXiv preprint arXiv:2009.12756*, 2020.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Renchi Yang. Efficient and effective similarity search over bipartite graphs. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 308–318, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3511959. URL https://doi.org/10.1145/3485447.3511959.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 535–546, 2021.

Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71:623–630, 2009. doi: 10.1140/epjb/e2009-00335-8.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. In *International Conference on Learning Representations*, volume 2026, pp. 147053–147075, 2026.

A ENTITY CANONICALISATION DETAILS

This appendix expands the canonicalisation pipeline summarised in Section 3.1.

NER backends. We extract named entities from each segment using the union of NuNER (Bogdanov et al., 2024) (209M parameters, zero-shot) and spaCy en_core_web_lg (Honnibal et al., 2020). NuNER produces better span merging for adjacent token sequences (especially for multi-word expressions and titles); spaCy contributes broader type coverage and a stable extraction on shorter mentions. Span-level deduplication merges overlapping extractions across the two backends: when two spans overlap, the longer span wins, with the type carried forward from whichever backend produced it (ties are broken in NuNER's favour because it is the higher-precision backend on multi-word entities).

Within-document short-to-long resolution. After span-level deduplication, we walk each document linearly and maintain a within-document map from short forms to longer forms by entity type. "Obama" and "Barack Obama" both PERSON map to one canonical PERSON; "Jordan" tagged LOC is not resolved to "Michael Jordan" tagged PERSON because the types differ. The by-type constraint avoids the most common false-positive collapse on Wikipedia text.

Title-anchor stage. Mentions are fuzzy-matched against the set of document titles in the corpus using a sequence similarity threshold of 0.8. When a mention exceeds the threshold against any document title, the entity is anchored to the canonical title form. This step leverages the convention in Wikipedia-derived corpora that article titles are the canonical entity names. On a corpus whose titles are not entity names the stage rarely fires, and global clustering carries most of the canonicalisation.

Global token-blocked clustering. The remaining surface-form variation is handled by token-blocked fuzzy clustering. Entities are first grouped into candidate blocks by shared tokens (so "Christopher Nolan" and "Nolan" share a block, but "Nolan" and "Akhenaten" do not). Within each block, pairs are scored by sequence similarity; pairs above 0.8 become edges in a similarity graph. Connected components in the graph define canonical clusters, with each cluster represented by its most frequent surface form. The token-blocking step reduces what would be an $O(n^2)$ pairwise comparison over n unique surface forms to approximately $O(n)$, since most blocks contain only a small number of variants.

Stored artefacts. The output of canonicalisation is a many-to-one map from surface mentions to canonical entities. The canonical entity set E is the unique image of this map, and forms the entity vertex set of the bipartite graph (Section 3.1, Equation 1).

B THE WALK ON A SINGLE QUERY

Figure 4 traces the walk of Section 3.2 on one 2WikiMultihopQA query.

C BASELINE, HARDWARE AND READER CONFIGURATION

This appendix gives the per-system configuration, hardware and reader details summarised in Section 4.

Lexical and dense. Dense retrieval uses `gte-Qwen2-1.5B-instruct` (Li et al., 2023) (L_2-normalised, with the model-card asymmetric query prompt applied to queries) and 3-sentence segments, ranking by cosine similarity. BM25 uses `rank_bm25` with default parameters ($k_1 = 1.5$, $b = 0.75$) over the same segments. RRF combines BM25 and dense with $k = 60$. The `+CE` suffix adds the same `ms-marco-MiniLM-L-6-v2` cross-encoder used by HEMLOCK.

ColBERTv2. The off-the-shelf `colbert-ir/colbertv2.0` checkpoint (Santhanam et al., 2022) via the `ragatouille` interface, in ColBERT's native indexing mode (documents split at its default 256 tokens). A controlled `colbertv2_segments` variant matching HEMLOCK's 3-sentence segmentation, ColBERT seed step and cross-encoder rerank is reported in Appendix F as a chunking-fairness check.

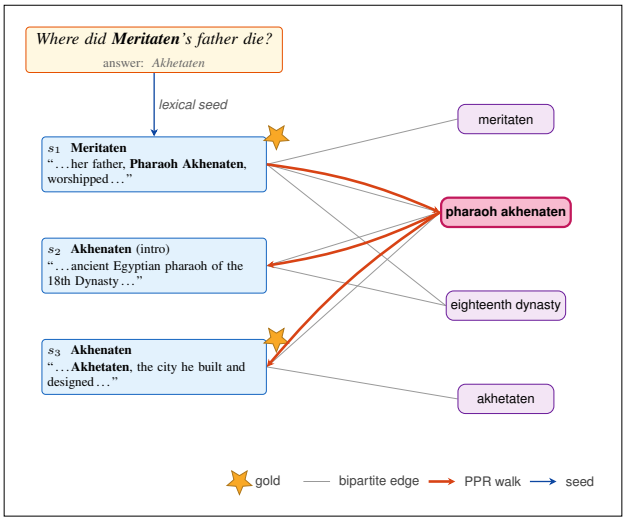


Figure 4: **What the graph walk does on a real query.** 2WikiMultihopQA q152, "Where did Meritaten's father die?" (answer: *Akhetaten*). Seeds reach s_1 but miss s_3; the walk crosses the bridge entity *Akhenaten* and surfaces it. It illustrates the mechanism; on this query ColBERTv2+CE also reaches recall@10 of 1.00.

LinearRAG. We run the released implementation on our corpora with the per-dataset hyperparameters from their `scripts/run.sh`, at both segment and document granularity, reporting the better.

IRCoT. `qwen3-8b` zero-shot at temperature 0, T=8 reasoning iterations (raised from the original T=4 to give iterative-LLM retrieval its strongest configuration), K=10 passages per iteration, dense retrieval as the base. Pricing follows `qwen3-8b` on OpenRouter as of May 2026 ($0.05/M input, $0.20/M output). Model-size and few-shot ablation in Appendix I.

HippoRAG 2. The PyPI `hipporag` package, which implements HippoRAG 2 rather than the original (Gutiérrez et al., 2024): LLM-based OpenIE triple extraction at index time, PPR over the extracted graph at retrieval time. The walk is not the whole query path: a per-query "recognition memory" step first has an LLM filter the top-ranked candidate triples, and the phrases surviving that filter are what seed the walk. The baseline therefore makes one generative call per query (their §3.4 and Appendix G.1). We use GPT-4o-mini (OpenAI, 2024) for OpenIE and NV-Embed-v2 (Lee et al., 2024) for embeddings. Their default filter model is Llama-3.3-70B-Instruct and we configure `gpt-4o-mini` in its place. Every per-query cost we quote for this baseline is therefore ours rather than theirs. Retrieval is at document granularity, and HEMLOCK's segments are deduplicated to unique documents before computing recall. Indexing time and dollar cost reflect API throughput.

Hardware and cost accounting. Retrieval measurements come from two machines: a single NVIDIA B200 (180GB SXM5, 20 CPU cores, 224GB RAM) for 2WikiMultihopQA, MuSiQue and IRCoT's local vLLM measurement, and an NVIDIA H200 (Xeon Platinum 8462Y+, 64 cores, 2015GB RAM) for HotpotQA dev, the per-query timings of Table 7, the LinearRAG comparison, the two-pass scoring measurements of Appendix J, and the indexing scaling runs. HippoRAG 2, IRCoT and the reader use the OpenRouter API. We report per-query and indexing wall-clock latency, and LLM call count as a hardware-independent cost proxy.

Reader. Each system's cached ranking is resolved to passage text and supplied to `qwen3-8b` through the OpenRouter API at temperature 0 with reasoning disabled, pinned to one provider; the served model and provider are recorded per generation and are constant across all 67,555 generations. We report SQuAD exact match and token-level F1 with paired bootstrap over identical query indices. Two control rows bound the comparison: `closed_book` supplies no context and measures what the reader already knows, and `gold` supplies the annotated gold context and measures what the reader

Table 7: **Indexing and per-query cost.** LLM calls and dollars are hardware-independent and comparable across every row; wall-clock is not, so we report it only within the locally-served class (HotpotQA dev, n=1000). IRCoT's figure is its local vLLM measurement.

System	Indexing cost	Per-query cost	Per-query wall-clock
Served locally — wall-clock comparable			
HEMLOCK	0 LLM calls, \$0	0 calls, \$0	96.0 ms[†]
ColBERTv2+CE	0 LLM calls, \$0	0 calls, \$0	24.5 ms
IRCoT (qwen3-8b, T=8)	none	1.5–2.3 calls, \$0.26–\$0.37/1k	430.0–650.0 ms[‡]
Served through a hosted API — wall-clock not comparable			
HippoRAG 2 (LLM-OpenIE)	6,193–21,100 calls, \$1.24–\$4.22	1 call, cost unmeasured	—

[†]Indexing is 5–15 min plus a 25 s PLAID pass. [‡]Local vLLM (`qwen3-8b`) on a B200 against HEMLOCK on an H200, n=100 HotpotQA, extrapolated per dataset by mean iterations; under the hosted deployment used for IRCoT's recall numbers the same queries take 17–29 s (Appendix I).

does with perfect retrieval. The reader is not offered an "unanswerable" option and always emits a best guess.

D Reconciling HEMLOCK's recall@10 figures

HEMLOCK's recall@10 appears at four combinations of split, graph variant and granularity, and Table 8 gives all of them for `ppr_colbert_raw_ce`, so any figure in the paper can be traced to one cell. The main text uses binary edges throughout: passage recall in Table 2 and Sections 5.1–5.5 (HotpotQA dev), and document recall where the comparison is against a system that retrieves documents (Table 3, HotpotQA train; Section 5.4, HotpotQA dev). Appendices E, G, H and L predate the binary graph: they are HotpotQA train on the weighted row, and comparisons within each are internally matched. Several of them also measure variants seeded from dense retrieval or from query entities. Those seeds need the 1.5B embedder that the recommended configuration does not carry, so those rows order the design choices rather than establish the numbers we report.

Table 8: **HEMLOCK recall@10 under every combination used in this paper.** Retriever is `ppr_colbert_raw_ce` throughout; 2WikiMultihopQA and MuSiQue have a single split.

	binary edges		weighted edges	
Corpus	passage	document	passage	document
HotpotQA train	0.9060	0.9220	0.9035	0.9220
HotpotQA dev	0.9120	0.9235	0.9045	0.9195
2Wiki	0.8754	0.8879	0.8739	0.8874
MuSiQue	0.7085	0.7285	0.7075	0.7250

E Segmentation paired bootstrap

Paired bootstrap of recall@10 on `ppr_dense_ce` between dynamic-programming geometric segmentation and uniform 3-sentence chunks: +2.6 pp on HotpotQA (p=0.002), +0.9 on 2Wiki (n.s.), and +1.2 on MuSiQue (p=0.046, marginal), all in favour of uniform chunking. An earlier unpaired comparison made the chunking lift look substantially larger on MuSiQue because the ablation subset (first 1,000 questions) is approximately 0.09 easier than the full dev set; pairing by query index removes the artefact. We adopt uniform chunking for plug-and-play simplicity rather than as a recall lift.

F ColBERTv2 chunking-fairness control

Table 2 reports ColBERTv2 in its native whole-document indexing mode, the configuration on which it ships its strongest results. To isolate the contribution of the graph walk from the contribution of

segmentation, we additionally evaluate ColBERTv2 over the same uniform 3-sentence segments used by HEMLOCK (`colbertv2_segments`). Holding segmentation, ColBERT seeding, and cross-encoder reranking constant, only the graph walk differs.

Table 9: **ColBERTv2 chunking-fairness control, recall@10.** `colbertv2_segments` runs ColBERTv2 over the same uniform 3-sentence segments as HEMLOCK, holding segmentation, ColBERT seeding, and cross-encoder rerank constant. HotpotQA is train, binary graph (Appendix D). Paired bootstrap, n_{iter}=10,000.

Dataset	`colbertv2_segments`	`ppr_colbert_raw_ce`	Δ	95% CI	p
HotpotQA	0.8305	0.9060	+0.076	[+0.062, +0.090]	$< 10^{-4}$
2Wiki	0.8063	0.8754	+0.069	[+0.056, +0.083]	$< 10^{-4}$
MuSiQue	0.6475	0.7085	+0.061	[+0.046, +0.077]	$< 10^{-4}$

Holding segmentation, seed step, and reranker constant, the walk adds $+6.1$ to $+7.6$ pp of recall@10 across the three multi-hop benchmarks. That margin runs above the $+4.2$ to $+5.4$ pp of Section 5.1 because `colbertv2_segments` is the weaker comparator, for the reason below; the figure we report is the one against native ColBERTv2+CE. `colbertv2_segments` runs slightly lower than native ColBERTv2 because late-interaction MaxSim favours ColBERT's larger native 256-token passages over the 3-sentence chunks; the headline retains native ColBERTv2 as the stronger baseline configuration.

G Composite scorer ablation and composite-versus-raw analysis

Composite scorer ablation. The composite scorer $\text{score}(s) = \beta_1 \cos(q, s) + \beta_2 \text{ent}(q, s) + \beta_3 g(s) + \beta_4 h(s)$ is load-bearing on multi-hop datasets where deeper bridge structure matters: dropping it costs -14 to -15 pp at recall@10 on MuSiQue and 2Wiki on PPR-based retrievers. It is optional on HotpotQA, where dropping it costs ≤ 2 pp and on `ppr_dense_ce` actually improves recall by $+1.2$ pp.

Composite-versus-raw with ColBERT seeds. For ColBERT-seeded retrievers the direction reverses: ranking the candidate union by raw PPR score beats the composite scorer on all three benchmarks (Table 10).

Table 10: **Composite versus raw rerank pool, recall@10.** ColBERT-seeded variants only. HotpotQA train, weighted graph (Appendix D).

Variant	**HotpotQA**	**2Wiki**	**MuSiQue**
`ppr_colbert_ce` (composite)	0.8715	0.8368	0.6765
`ppr_colbert_raw_ce` (raw)	0.9035	0.8739	0.7075
Δ (raw − composite)	+0.032	+0.037	+0.031

ColBERT scores already encode strong query-passage similarity, and the composite's $\beta_1 \cos + \beta_2 \text{ent}$ terms dilute it. Ranking by PPR score alone leaves the candidate cut ordered by graph evidence, which is what Appendix M measures. HEMLOCK therefore uses `raw`.

H Graph-walk operator: PPR vs BFS_2

We compare the PPR walk used in HEMLOCK against a depth-bounded BFS_2 alternative that takes the union of the seed nodes and their 2-hop entity-segment neighbours. Both operators consume identical seed sets and are scored under the same composite reranker + cross-encoder pipeline. The two walks differ only in how they propagate seed mass: BFS_2 applies a hard depth-2 cutoff, while PPR computes a steady-state distribution over the full bipartite graph with damping $\alpha = 0.85$.

Table 11 reports paired-bootstrap deltas (passage recall@10, n=1000 per dataset, 10,000 iterations) for four seed-source variants. With dense seeds, BFS_2 matches or modestly exceeds PPR on all

three; with entity seeds, PPR wins on two of three. With ColBERTv2 seeds — the configuration used in HEMLOCK's headline — PPR outperforms BFS_2 on every dataset under both composite and raw scoring (margins $+0.7$ to $+7.5$ pp), significantly on all three under raw scoring and on two of three under composite scoring (HotpotQA p=0.14). The operator-choice gap widens as seed quality improves: PPR's full-graph distribution amortises over the larger neighbourhoods that strong seeds expose, where BFS_2's hard cutoff truncates relevant evidence. We adopt PPR for HEMLOCK and treat BFS_2 as a lighter-weight alternative when ColBERT-class indexing is unavailable (recall-competitive with PPR under dense seeds on all three). Recall@10 at BFS depths 2, 3, and 4 is within 0.2 pp on the segment projection graph, so 2-hop reach suffices.

Table 11: **Graph-walk operator ablation: PPR vs BFS_2** at matched seeds and scoring (passage recall@10, paired bootstrap, 10,000 iterations). Δ is PPR mean $-$ BFS_2 mean on shared q_idx; p is the one-sided bootstrap probability that PPR does not improve over BFS_2. Bold indicates $p < 0.05$ in the direction of PPR. HotpotQA train, weighted graph (Appendix D).

Dataset	Seeds + scoring	BFS_2	PPR	Δ	95% CI	p
Dense seeds + composite + CE						
HotpotQA	dense	0.8385	0.8250	−0.014	[−0.027, −0.001]	0.98
2Wiki	dense	0.7533	0.7217	−0.032	[−0.051, −0.012]	1.00
MuSiQue	dense	0.6370	0.5715	−0.066	[−0.083, −0.049]	1.00
Entity seeds + composite + CE						
HotpotQA	entity	0.8205	0.8195	−0.001	[−0.019, +0.016]	0.55
2Wiki	entity	0.7432	0.8879	**+0.145**	[+0.129, +0.161]	**$< 10^{-4}$**
MuSiQue	entity	0.6045	0.6280	**+0.024**	[+0.002, +0.045]	**0.017**
ColBERTv2 seeds + composite + CE						
HotpotQA	colbert	0.8650	0.8715	+0.007	[−0.005, +0.018]	0.14
2Wiki	colbert	0.7618	0.8368	**+0.075**	[+0.061, +0.090]	**$< 10^{-4}$**
MuSiQue	colbert	0.6630	0.6765	**+0.013**	[+0.001, +0.027]	**0.022**
ColBERTv2 seeds + raw + CE (HEMLOCK headline configuration)						
HotpotQA	colbert (raw)	0.8355	0.9035	**+0.068**	[+0.056, +0.081]	**$< 10^{-4}$**
2Wiki	colbert (raw)	0.8428	0.8739	**+0.031**	[+0.021, +0.041]	**$< 10^{-4}$**
MuSiQue	colbert (raw)	0.6475	0.7075	**+0.060**	[+0.047, +0.074]	**$< 10^{-4}$**

I IRCoT ROBUSTNESS ABLATION

Latency methodology. IRCoT recall numbers are measured against OpenRouter (n=1000 per dataset) since that is the deployment most users will encounter. For latency comparison with the locally measured retrievers, we additionally measure IRCoT under qwen3-8b served via local vLLM on an NVIDIA B200, with identical temperature and stop-condition settings; deterministic top-1 sampling guarantees recall numbers match across endpoints. We sample n=100 HotpotQA queries and extrapolate per-dataset latencies by scaling the HotpotQA local measurement by each dataset's empirical mean iteration count (Table 12). The OpenRouter deployment adds ∼15–30 s/query of network and queueing overhead; we report it as deployment latency, not the compute-fair comparison. Local recall at n=100 runs slightly higher than the n=1000 OpenRouter numbers in the headline due to sampling variance and possible OpenRouter quantisation differences; we use OpenRouter recall for the headline (larger n) and local measurements for compute-fair latency only.

*Extrapolated by per-iteration constant ($0.43/1.80 \approx 0.24$ s/iter) times each dataset's mean iteration count.

Model size and few-shot ablation. We ablate IRCoT along two axes on a 300-question HotpotQA subset: LLM size (qwen3-8b vs qwen3-32b) and prompting (zero-shot, 5-shot, 10-shot). Few-shot demonstrations are hand-crafted Wikipedia-style multi-hop exemplars including 2 gold and 2–3 distractor passages each, mirroring inference-time retrieval conditions; demonstration 6 shows the "insufficient context, request next sub-question" pattern from Trivedi et al. (2023). Demonstrations are constructed independently of any benchmark's train/dev/test split.

Table 12: **IRCoT cost across the three datasets.** Wall-clock under OpenRouter is the deployment number; wall-clock under local vLLM is the compute-fair number used in Table 7. Local per-dataset values for 2Wiki and MuSiQue are extrapolated from the HotpotQA n=100 measurement using each dataset's mean iteration count.

Dataset	LLM calls/q	$/1k q	OpenRouter s/q	Local vLLM s/q
HotpotQA	1.52	0.260	21.15	0.43
2Wiki	1.99	0.362	29.17	0.56*
MuSiQue	2.29	0.368	28.94	0.65*

Table 13: **IRCoT ablation on HotpotQA** (n=300, paired with HEMLOCK by q_idx). Two axes: LLM size (qwen3-8b vs qwen3-32b) and number of in-context demonstrations. Δ vs HEMLOCK is the paired-bootstrap mean of `ppr_colbert_raw_ce` minus IRCoT recall@10 on the same q_idx subset; p is one-sided p(HEM≤IR) from 10,000 bootstrap samples. Local s/q is IRCoT's latency under local vLLM on a B200; HEMLOCK's is its H200 retrieval latency, so the two are not same-machine. OpenRouter s/q is the deployment latency.

Configuration	r@10	mean iters	Local s/q	OpenRouter s/q	$/1k q	Δ vs HEMLOCK (p)
8B zero-shot (n=1000)	0.758	1.52	0.43	21.2	0.260	+0.146 (p<0.0001)
8B + 5-shot	0.825	1.19	0.26	13.5	0.266	+0.065 (p<0.0001)
8B + 10-shot	**0.837**	1.54	—	21.3	0.469	+0.053 (p=0.0009)
32B zero-shot	0.800	5.58	—	10.4	0.455	+0.090 (p<0.0001)
32B + 5-shot	0.805	5.77	—	10.4	0.775	+0.085 (p<0.0001)
HEMLOCK (`ppr_colbert_raw_ce`)	0.890	—	0.096	—	0	—

The strongest IRCoT cell is qwen3-8b with 10 demonstrations (`r@10`= 0.837), which improves on the zero-shot 8B baseline by +7.9 pp but remains +5.3 pp below HEMLOCK (p=0.0009). Increasing model scale alone yields a smaller lift: qwen3-32b zero-shot (`r@10`= 0.800) improves on 8B zero-shot by only +4.2 pp, and does not benefit appreciably from few-shot prompting (32B 5-shot 0.805 ≈ 32B zero-shot 0.800). qwen3-32b averages 5.6 iterations per query versus 1.5 for qwen3-8b, suggesting either deeper reasoning chains or less reliable termination on the "answer is" stop phrase; either interpretation leaves the IRCoT-vs-HEMLOCK gap above 5 pp across all five configurations. qwen3-32b + 5-shot is Pareto-dominated by qwen3-8b + 5-shot — worse `r@10` (0.805 vs 0.825) at higher per-query cost ($0.775 vs $0.266 per 1k queries) — indicating that within IRCoT, prompting matters more than model scale at the 8B–32B range, and that scaling and prompting are not additive on this task.

n=1000 **follow-up: 5-shot does not replicate at scale.** At n=1000 across all three Wiki datasets, the +6.7 r@10 improvement of 8B+5-shot over 8B zero-shot does not replicate: 8B+5-shot recall is 0.7455 on HotpotQA, 0.7152 on 2Wiki, 0.5610 on MuSiQue, within paired-bootstrap noise of 0-shot on each (−1.3, +0.3, −2.0 pp respectively). The n=300 lift was a subsample artefact: few-shot demonstrations make qwen3-8B terminate earlier (mean iterations 1.29–1.88 for 5-shot vs 1.52–2.29 for 0-shot), which helps on the easier first-300 questions but not at scale. HEMLOCK's lead over IRCoT remains at least +14.0 pp of recall@10 across the configurations tested at this scale (Table 3). The headline IRCoT row in Table 3 reports 8B zero-shot at n=1000, which matches the prompting setup of Trivedi et al. (2023) and outperforms 8B+5-shot at scale on 2 of 3 Wiki datasets.

J THE TWO-PASS REDUCTION: EXPANSION AND IDENTITY CHECKS

This appendix expands Section 3.3. Let $G = (S \cup E, W)$ be the bipartite graph of Section 3.1 and M its column-stochastic transition matrix, so that $M_{es} = w(s,e)/\deg_S(s)$ and $M_{s'e} = w(s',e)/\deg_E(e)$. PPR with restart probability ρ and personalisation vector r supported on $\mathcal{S}_0$ is the Neumann series

$$\pi = \rho \sum_{t \geq 0} (1-\rho)^t M^t r. \tag{4}$$

Because G is bipartite and r is supported entirely on segments, $M^t r$ is supported on entity nodes for odd t and on segment nodes for even t. Restricting (4) to S therefore annihilates every odd term, and the first two surviving terms are $t{=}0$ (the seeds themselves) and $t{=}2$:

$$\pi|_S = \rho\, r + \rho(1-\rho)^2 M^2 r|_S + O\big((1-\rho)^4\big)\,, \qquad M^2 r|_S = \mathrm{CF}, \tag{5}$$

with CF as in Equation 3: the first pass accumulates $a(e)$ over the seeds' incident entities, the second redistributes it to candidate segments under the $1/\deg_E$ discount. Two steps compose to the segment-to-segment kernel $P(s,s') = \sum_e w(s,e)\, w(s',e)/(\deg_S(s)\deg_E(e))$, with $M^2 r|_S = P^\top r$. Summing every even term instead of truncating gives the restriction in closed form,

$$\pi|_S = \rho\left(I - (1-\rho)^2 P^\top\right)^{-1} r, \tag{6}$$

of which (5) is the first order in P and is what we deploy.

The projected operator. Writing $U(s,e) = w(s,e)/\deg_S(s)$ and $V(e,s') = w(s',e)/\deg_E(e)$ for the two normalisations, $P = UV$ is the hidden transition matrix of Deng et al. (2009), stated as Equation 2 of Yang (2022) and Equation 3 of Liu & Luo (2024) under the identification of their two node sets with S and E. PPR over P is Hidden PPR, their Equations 3 and 4. Matching coefficients, $\pi|_S$ is Hidden PPR at $\alpha = 1-(1-\rho)^2 = 0.2775$ up to the positive scalar ρ/α, and CF is its one-hop term, Equation 5 of Liu & Luo (2024). Both papers call the one-directional measure biased and replace it with its symmetrised form. We keep the one-directional score, because the walk runs from seeds to candidates. Materialising P is what makes Hidden PPR expensive: it can carry $O(|S|^2)$ non-zero entries, since one entity appearing in n segments contributes n^2 segment pairs, and on HotpotQA dev that bound is $17{,}131^2 \approx 2.9 \times 10^8$. The two passes never form it, walking instead the incidence lists of the k_0 seeds and of the entities those seeds reach, so cost scales with edges touched. That saving is an implementation choice rather than a consequence of the identity.

Truncation error. At $\rho = 0.15$ the two retained terms of (5) carry 0.150 and 0.108 of the segment mass, against 0.282 in the discarded tail. The remainder is larger than either term kept, so the truncation does not rest on the remainder being small. It nonetheless preserves the candidate set: CF decides which 30 segments reach the cross-encoder, the cross-encoder re-sorts those 30, and the higher-order terms mostly reorder within the set. Over that same pool, measured against the converged walk, CF reaches Spearman correlation 0.93–0.96, Kendall τ of 0.82–0.87, and top-ten overlap of 0.89–0.91, with recall@10 unchanged (below).

The degree discount. Zhou et al. (2009) give the resource-allocation score in two forms: symmetric over common neighbours (their Equation 10), and asymmetric with a further division by the source degree (their Equation 11). On binary edges CF from a single seed is Equation 11, with $1/\deg_S(s)$ in the role of their $1/k(x)$ and the shared neighbours being entities rather than nodes of a single mode. Their graph is unweighted and simple. The weighted variant of Section 3.1 therefore departs from it: the transition matrix normalises by $\sum w$, so degrees there are weighted sums rather than counts. On the binary graph the two coincide.

Identity checks. Four checks per dataset. Each compares the implementation against the algebra above; the comparison with the converged walk is the rank agreement and the recall reported either side of them. The one-step image Mr has zero mass on segments, confirming that no segment-segment edge entered the graph (exactly 0 on all three). The two-pass computation, which accumulates over incidence dictionaries and never materialises M, agrees with $M^2 r$ to a maximum relative error of $3.99{\times}10^{-16}$, $3.78{\times}10^{-16}$, and $4.62{\times}10^{-16}$. Computing the $t{\le}2$ truncation directly from M and comparing it against $\rho r + \rho(1-\rho)^2$ CF agrees to $3.04{\times}10^{-16}$, $1.82{\times}10^{-16}$, and $3.05{\times}10^{-16}$. And on 50 sampled queries per dataset CF selects the same top-30 candidate set as that truncation, $50/50$ on each.

That last check first returned $45/50$. All five disagreements turned out to be the identical top-30 set in permuted order, caused by floating-point summation order breaking exact ties. Ties are common on the binary graph, where only 61% of top-50 scores are distinct, so the order inside the cut is partly arbitrary. The cross-encoder re-sorts those 30, which is why the check is on set equality.

Recall and scoring cost. HotpotQA figures in this paragraph are the training split; Section 3.3 uses the dev values of Table 6, and 2WikiMultihopQA and MuSiQue are identical in both. Here $\rho = 0.15$, one minus the damping factor of Section 3.2. Ranking by CF reproduces the converged walk's recall@10: 0.9065, 0.8754, and 0.7085 against 0.9060, 0.8754, and 0.7085. The walk takes about 89 matrix-vector products per query and the two passes two. Removing the restart term leaves recall unchanged to four decimal places. Ranking by the resource-allocation score alone stays within 0.5 pp of the walk (0.9035, 0.8769, 0.7040); removing its $1/\deg_E$ discount lowers recall@10 to 0.8440, 0.8343, and 0.6560. On the weighted variant of Section 3.1 the same reduction holds, at 0.9045, 0.8779, and 0.7060 against that walk's 0.9035, 0.8739, and 0.7075. On the H200 of Section 4, the scoring stage falls from 32 to 59 ms per query for the walk to 1.3 to 3.2 ms for the two passes, 17.8 to 25.7× faster, and below 0.6 ms for the resource-allocation score. The cross-encoder is unaffected, and we make no end-to-end latency claim.

K LINEARRAG: FULL COMPARISON

This appendix expands Section 5.4. LinearRAG reports answer accuracy rather than recall, so no published number of theirs is reproduced here. As a harness check, on their own released 2WikiMultihopQA corpus we find the gold answer within the top five for 78.5% of queries against their reported 70.2% containment accuracy, and their graph beats their own dense fallback by +34.1 pp at k=5. Their released corpora are far coarser than ours (2WikiMultihopQA: 658 passages averaging ∼4,400 characters, against our 10,103 segments averaging 267), which is why all recall here is document-level.

Non-transitivity with their reported margins. LinearRAG reports containment accuracy, not recall: whether an answer string appears anywhere in the retrieved text. At a fixed k that quantity rewards long passages, and theirs average ∼4,400 characters against our 267, so a system retrieving fewer, larger passages can report higher containment while retrieving less gold. Their comparison also runs every system, HippoRAG 2 included, with all-mpnet-base-v2 (110M) in place of HippoRAG 2's released NV-Embed-v2 (7B); we keep NV-Embed-v2. This is why both systems are normalised to document level here, and why their reported result against HippoRAG 2 and ours against LinearRAG cannot simply be chained.

Their operator under the same reduction. Their walk differs from ours in three ways that bear on it. The graph is not bipartite: passage-entity edges are supplemented by a passage-passage adjacency chain. Restart mass sits on entity nodes rather than on passages. The entity-mediated term therefore appears at first order rather than second, and the discount enters as $1/\deg_E$ rather than $1/\deg_E^2$. Damping is 0.5 in their released code, against the 0.85 their paper states.

Expanding their walk as a Neumann series (Appendix J) and restricting to passage nodes yields the same degree-discounted score. Our reconstruction of their operator returns their implementation's own top-ten document set exactly on all three corpora (200 queries each). Ranking by the first-order entity term alone agrees with their converged walk on 64.5, 50.4, and 68.2% of the top ten; removing the degree discount drops that to 34.9, 36.3, and 22.2%. Two independently designed systems instantiate the same operator, and in both the discount carries it. Their collapse loses more than ours, because the passage chain breaks bipartiteness and their odd-order terms do not vanish: two matrix-vector products recover 93.8, 85.6, and 95.0% of their converged ranking, three recover 96.5, 93.6, and 96.7%. These figures are measured at their published λ.

Recall per token. Table 14 gives the per-budget deltas behind Figure 1.

Budgets above 2000 are omitted: HEMLOCK carries ∼2,400–2,550 tokens at depth 30 and is exhausted there, so those rows measure list depth rather than retrieval. At 4000 the remaining cells are ties or reverse. LinearRAG's units run 137.0, 144.9, and 115.9 tokens at its published λ against HEMLOCK's 84.8, 82.3, and 86.3.

Matched granularity. Their released corpora are coarser than our segments, so the per-token comparison could be read as a chunking artefact. We therefore re-ran LinearRAG's own pipeline over our 3-sentence segments, at its published hyperparameters and with both ranked lists depth-matched

Table 14: **Document recall at matched input-token budgets**, HEMLOCK minus LinearRAG, both at a single fixed configuration and depth-matched at 30. LinearRAG is at document granularity and its published λ. All cells $p \leq 2 \times 10^{-4}$.

Budget (tokens)	HotpotQA	2Wiki	MuSiQue
100	+0.1791	+0.1348	+0.1805
250	+0.1976	+0.1059	+0.1830
500	+0.1596	+0.0847	+0.1625
1000	+0.1110	+0.0510	+0.1281
2000	+0.0520	+0.0335	+0.0536

at 30, which leaves the two systems at 71–79 and 82–86 tokens per retrieved unit. HEMLOCK still retrieves more gold per input token at every budget from 100 to 2,000 tokens on all three benchmarks (Table 15; all $p < 0.05$, and $p < 10^{-3}$ except the 1,000- and 2,000-token rows on 2WikiMultihopQA and the 2,000-token row on MuSiQue). Margins narrow at matched granularity, as expected, but do not close. HotpotQA is its train split here, because their segment corpus is built from it, and at 2,000 tokens both lists start to run out (9–35% of LinearRAG's and 1–6% of ours). The rows at or below 1,000 tokens are the clean ones.

Table 15: **Document recall at matched input-token budgets with LinearRAG re-indexed on our 3-sentence segments**, HEMLOCK minus LinearRAG, both depth-matched at 30 and at LinearRAG's published hyperparameters. HotpotQA is train.

Budget (tokens)	HotpotQA	2Wiki	MuSiQue
100	+0.1521	+0.0918	+0.1550
250	+0.1466	+0.0553	+0.1575
500	+0.1290	+0.0376	+0.1240
1000	+0.1036	+0.0230	+0.0966
2000	+0.0676	+0.0200	+0.0206

Exact match under the frozen reader. Table 16 supplies both systems' rankings to the frozen reader of Section 5.3 at equal k. LinearRAG's units are whole documents, so at the same k its prompts run 1.4–1.6× longer than HEMLOCK's. LinearRAG is never significantly ahead; HEMLOCK is significantly ahead in three of the nine cells, and all three stay below the Bonferroni threshold for nine tests ($p < 0.0056$). From k=5 to k=20 LinearRAG gains +9.3, +0.2 and +1.2 pp and HEMLOCK gains +4.7, +2.8 and +7.3. LinearRAG's exact match stays flat on 2WikiMultihopQA and MuSiQue as k grows.

Table 16: **Exact match against LinearRAG under the frozen reader**, equal k, LinearRAG at its published hyperparameters. HotpotQA is dev. p is one-sided, $p(HEMLOCK \leq \text{LinearRAG})$; n.s. where $\Delta < 0$. *Tokens* is the mean prompt tokens per question as billed by the reader, LinearRAG / HEMLOCK.

Benchmark	k	LinearRAG	HEMLOCK	Δ	95% CI	p	Tokens
HotpotQA	5	0.3870	0.4380	**+0.0510**	[+0.022, +0.080]	0.0002	808 / 588
	10	0.4320	0.4780	**+0.0460**	[+0.020, +0.072]	0.0001	1,559 / 1,075
	20	0.4800	0.4850	+0.0050	[−0.020, +0.030]	0.370	3,122 / 2,086
2Wiki	5	0.2713	0.2653	−0.0060	[−0.034, +0.022]	n.s.	863 / 580
	10	0.2723	0.2793	+0.0070	[−0.020, +0.033]	0.309	1,609 / 1,069
	20	0.2733	0.2933	+0.0200	[−0.006, +0.047]	0.071	3,290 / 2,098
MuSiQue	5	0.1520	0.1350	−0.0170	[−0.041, +0.007]	n.s.	852 / 591
	10	0.1620	0.1740	+0.0120	[−0.012, +0.035]	0.170	1,530 / 1,086
	20	0.1640	0.2080	**+0.0440**	[+0.020, +0.067]	0.0003	2,833 / 2,109

Read at LinearRAG's own per-question token budget instead of at equal k, HEMLOCK's exact match is at or above LinearRAG's in all nine cells: +6.9, +4.9 and +0.5 pp on HotpotQA, +0.2, +1.4 and

+2.0 on 2WikiMultihopQA, and +0.4, +2.7 and +4.4 on MuSiQue, at k=5, 10 and 20. These are read off HEMLOCK's exact-match curve between adjacent k; at k=20 LinearRAG's budget exceeds HEMLOCK's largest measured k, so the equal-k value stands. They are estimates from mean exact match and carry no confidence intervals.

Robustness to λ. Table 5 reports LinearRAG at the per-dataset hyperparameters it ships. One of those, the dense-mixing weight λ, is arguably mis-set on our corpora: it weights a dense channel whose embedder truncates at 384 tokens, while their released passages have a median length of 1,004. Only 38.2% of each passage is ever embedded, and λ was fitted in that regime. Sweeping λ on our corpora raises them by 4.2 to 6.6 pp of recall@10, to 0.9150, 0.9039, and 0.6915. We do not report that as the headline comparison, because it is a search over the evaluation corpora that we did not perform for HEMLOCK; granting it to one system and not the other would not be a controlled comparison. Under it HEMLOCK still leads on MuSiQue (+3.7 pp, p=0.0003), is level on HotpotQA (+0.9, p=0.123), and trails on 2WikiMultihopQA (−1.6). The token budgets of Table 14 are likewise reported at their published λ; at the swept λ their units shorten to 98–125 tokens and HEMLOCK's lead narrows but holds through 500 tokens on all three.

Appendix L bounds the equivalent search for HEMLOCK: choosing the best of fifteen graph variants per dataset would gain at most 0.6 pp of document recall@10, against the 4.2–6.6 that tuning λ is worth to LinearRAG. The damping factor is the canonical PageRank value and was not tuned.

Cost. LinearRAG reports 0.093 s average retrieval time across its benchmark suite, close to our own measurement of HEMLOCK; we report both their published figure and what we measure on our corpora, since the two differ. The gap in our measurement widens on MuSiQue, where their passage scoring iterates over every passage and activated entity. For indexing, LinearRAG's published times at 5M and 10M tokens imply a scaling exponent of 1.129, against HEMLOCK's 1.045 (Section 5.6); that is a ratio of two points on other hardware, so we compare exponents rather than wall-clock.

L TUNING SYMMETRY

Section 5.4 sweeps LinearRAG's λ on our corpora and reports its best configuration. This appendix bounds what the equivalent search would have been worth to HEMLOCK, so the asymmetry can be read in both directions.

Across fifteen HEMLOCK graph variants evaluated on the same questions, Table 17 compares the per-dataset best against the single configuration we commit to throughout (`ppr_colbert_raw_ce`), at document granularity for comparability with Section 5.4.

Table 17: **What per-dataset configuration selection would buy HEMLOCK.** Document recall@10 over fifteen graph variants on the same questions. *Gain* is the per-dataset best minus the fixed configuration; *rank* is the fixed configuration's position among the fifteen. HotpotQA train, weighted graph (Appendix D).

Benchmark	per-dataset best	`ppr_colbert_raw_ce`	gain	rank
HotpotQA	0.9220	0.9220	0.0000	1/15
2Wiki	0.8934 (`ppr_entity_ce`)	0.8874	+0.0060	2/15
MuSiQue	0.7250	0.7250	0.0000	1/15

Committing a priori to one configuration costs at most 0.6 pp, against the 4.2–6.6 pp of recall@10 that sweeping one hyperparameter is worth to LinearRAG. `ppr_colbert_raw_ce` is also the best *fixed* configuration across all three benchmarks, so the choice is stable rather than a per-dataset selection.

This measures configuration selection, not the whole design search: the fifteen variants were themselves designed and filtered across three iterations on these questions, and losing families were dropped after their numbers were seen. The claim is that the final selection is cheap and stable, not that no information from these benchmarks entered the architecture.

M THE CANDIDATE CUT: CONTROLS AND ROBUSTNESS

This appendix expands the candidate-cut analysis of Section 5.5. Table 18 gives the four ways of reducing the candidate union to the reranker's input, and the availability and selection decomposition quoted there.

Table 18: **Passage recall@10 under four ways of reducing the same candidate union to the reranker's input.** *Availability* is the gain from ranking the whole union over cutting by ColBERTv2 score (which returns ColBERTv2's own top-30); *selection* is the further gain from cutting by PPR score. Random cuts are means of repeated draws. HotpotQA is dev.

Benchmark	**random** 30	**ColBERTv2-cut**	**all** $\sim$51	**PPR-cut**	**avail.**	**select.**
HotpotQA	0.6043 ± 0.011	0.8705	0.8870	**0.9120**	+0.0165	+0.0250
2Wiki	0.5863 ± 0.009	0.8213	0.8318	**0.8754**	+0.0105	+0.0436
MuSiQue	0.4609 ± 0.009	0.6635	0.6665	**0.7085**	+0.0030	+0.0420

The decomposition merges cached cross-encoder score lists rather than rerunning the reranker. Cross-encoder scores are bit-identical across runs (maximum difference 0.000 over the $\sim$27,000 candidates scored in both). The merge reproduces what the reranker would output, and the simulation is exact.

Random-cut control. Cutting the same union to 30 at random collapses recall to 0.604, 0.586, and 0.461, 25 to 31 pp below the PPR cut and below applying no cut at all. The gain therefore comes from the PPR ordering rather than from the smaller pool.

Rank-fusion cut. Truncating the union by reciprocal rank fusion of ColBERTv2 and PPR ranks scores 0.8740 on HotpotQA dev, 3.1 pp below the PPR cut on the same index (95% CI $[-4.4, -1.8]$; 51 queries better, 115 worse), and equal to ColBERTv2+CE there. This run uses a rebuilt ColBERTv2 index, on which ColBERTv2+CE scores 0.8740 and the PPR cut 0.9050; a PLAID rebuild moves both by a few thousandths.

Per-query gains and losses. Against ColBERTv2+CE, HEMLOCK gains recall on 119, 160, and 172 queries and loses it on 50, 55, and 83. Every gold segment it loses was in ColBERTv2's top-30, and so in the union, and was removed by the PPR truncation before the cross-encoder scored it.

Robustness to reranker capacity. These measurements use `ms-marco-MiniLM-L-6-v2`. Rescoring identical pools with `ms-marco-MiniLM-L-12-v2` (33M parameters) leaves the pattern intact: promotion of graph-only gold segments into the top ten moves from 42.8% to 42.3% on 2WikiMultihopQA and from 28.7% to 29.5% on MuSiQue, and recall@10 moves by at most 0.7 pp with no consistent sign. A deeper cross-encoder from the same family leaves the deficit open; a reranker trained on different data closes part of it (Section 5.5).

Pool size. The merged pool is ColBERTv2's 30 united with the 30 surviving the cut, about 51 candidates. The $\sim$9 expansion candidates below the cut were scored by neither run, and including them would enlarge the pool further, so the measured penalty is likely an underestimate.

N FURTHER EVIDENCE THAT THE BOTTLENECK IS RANKING

This appendix expands the ranking analysis of Section 5.5.

Depth and answer accuracy. Exact match increases through k=10 on all three benchmarks and, on the binary graph, through k=20 as well (+2.8, +1.4, +3.4 pp over that last interval; Section 5.3). Evidence the reader can use is already present below rank ten.

Larger candidate budget. Raising d from 30 to 100 raises recall@10 by +1.0 pp on HotpotQA dev. Neither the seeds nor the graph change; the union grows, the truncation keeps 100 candidates rather than 30, and the reranker's pool grows with it. We report d=30 throughout.

Documents the walk cannot reach. Of the gold documents missing from ColBERTv2's top 30, 9.4, 13.9, and 17.4% share no entity with any seed and are unreachable by the walk at any depth, and a further 8.3, 7.7, and 8.4% share only generic entities with the seeds (Figure 3).

Reranker family. Capacity within one training family does not help: `ms-marco-MiniLM-L-12-v2` (33M) leaves promotion of graph-only gold within 2 pp on every benchmark, while `bge-reranker-base` (278M, different training data) pulls its median rank from 9, 13, and 15 to 3, 9, and 13. Rescoring ColBERTv2+CE's pool with the same reranker isolates whether the gain belongs to the graph or to the reranker (Table 19).

Table 19: **The graph's contribution under a stronger reranker.** Identical candidate pools rescored with `ms-marco-MiniLM-L-12-v2` (33M, same training data) and `bge-reranker-base` (278M, different data) in place of `ms-marco-MiniLM-L-6-v2` (22M). Both are controls, not HEMLOCK configurations; the MiniLM-L-6 rows reproduce Table 2.

Benchmark	Reranker	HEMLOCK	ColBERTv2+CE	Graph contribution
HotpotQA	MiniLM-L-6	0.9120	0.8705	+0.0415
	MiniLM-L-12	0.9050	0.8705	+0.0345
	BGE-base	0.9365	0.8905	**+0.0460**
2Wiki	MiniLM-L-6	0.8754	0.8213	+0.0541
	MiniLM-L-12	0.8759	0.8208	+0.0551
	BGE-base	0.8834	0.8158	**+0.0676**
MuSiQue	MiniLM-L-6	0.7085	0.6635	+0.0450
	MiniLM-L-12	0.7095	0.6710	+0.0385
	BGE-base	0.7290	0.6620	**+0.0670**

The graph's contribution survives the swap and grows, by +0.5, +1.4, and +2.2 pp. BGE-base helps HEMLOCK's pool while slightly *hurting* ColBERTv2+CE's on 2WikiMultihopQA and MuSiQue. The extra candidates the graph supplies are what a stronger reranker has to work with.

Cost of the reranker swap. `bge-reranker-base` raises recall@10 over the same candidate pool by 2.5, 0.8, and 2.1 pp, and that gain carries a cost: the same swap loses 2.0 and 4.5 pp of promotion on the gold ColBERTv2 had already found, on 2WikiMultihopQA and MuSiQue. The deeper MS MARCO model moves the graph's contribution by −0.7, +0.1, and −0.7 pp, with no consistent sign.

O THE NAMED-EVIDENCE GRADIENT

Table 20 gives the per-stratum figures behind Section 5.1. The rule reads gold documents and stratifies evaluation rather than routing queries. Excluding 2WikiMultihopQA's comparison questions, the contribution there is +11.3 pp.

P FULL EXACT-MATCH GRID

Table 21 gives every exact-match value; Figure 5 plots them against k and Figure 6 against reader input tokens, with every baseline. On HotpotQA dev the reader's absolute scores fall (HEMLOCK 0.4780 at k=10, gold-context ceiling 0.5940), but the ordering is unchanged at every k: HEMLOCK leads RRF+CE by +3.8 pp at k=10 and +4.7 at k=20, and ceiling-normalised capture is 76.8% against 75.9% on train. At k=3 the margin over RRF+CE is at most +1.2 pp and reaches significance only on 2WikiMultihopQA (p=0.042). Per added passage, both cross-encoder baselines decline on 2WikiMultihopQA at every interval, while HEMLOCK gains +0.1 and +1.4 pp over 3→5 and 5→10. On MuSiQue, seventeen additional passages move RRF+CE by +0.2 pp EM while HEMLOCK gains +8.5. At k=20 HEMLOCK captures 80.3% and 43.7% of the closed-book-to-gold range on HotpotQA and MuSiQue.

Table 20: **The seed-matched graph contribution by how much evidence the query names.** Passage recall@10 on the binary graph; *base* is ColBERTv2+CE and Δ is HEMLOCK minus base on the same questions, paired bootstrap n_{iter}=10,000. HotpotQA is dev. Every 2WikiMultihopQA question names at least one gold document.

Benchmark	Question names	n	base	HEMLOCK	Δ (95% CI)
HotpotQA	neither gold document	313	0.8898	0.9073	$+0.0176$ $[-0.0048, +0.0399]$
	one gold document	601	0.8453	0.9068	$\mathbf{+0.0616}$ $[+0.0441, +0.0790]$
	both gold documents	86	0.9767	0.9651	-0.0116 $[-0.0407, +0.0116]$
2Wiki	one gold document	660	0.7394	0.8530	$\mathbf{+0.1136}$ $[+0.0970, +0.1311]$
	both gold documents	339	0.9808	0.9189	$\mathbf{-0.0619}$ $[-0.0796, -0.0457]$
MuSiQue	neither gold document	416	0.6575	0.6611	$+0.0036$ $[-0.0204, +0.0264]$
	one gold document	543	0.6464	0.7302	$\mathbf{+0.0838}$ $[+0.0617, +0.1050]$
	both gold documents	41	0.9512	0.9024	$\mathbf{-0.0488}$ $[-0.0976, -0.0122]$

Table 21: **Exact match under a frozen reader, full grid**, varying retrieval depth k. Prompt, decoding and reader are identical across rows; only the retrieved passages differ. Closed-book and gold-context bounds are given per dataset. IRCoT and HippoRAG 2 call an LLM and are not measurable on MuSiQue or beyond k=10. HotpotQA is train.

System	k=3	k=5	k=10	k=20
HotpotQA (closed-book 0.1380, gold 0.7760)				
Dense	0.4190	0.4580	0.4990	0.5350
RRF (BM25+Dense)+CE	0.5120	0.5360	0.5710	0.5770
ColBERTv2+CE	0.5080	0.5350	0.5440	0.5690
HEMLOCK	0.5190	0.5640	**0.6220**	**0.6500**
IRCoT (frozen reader)	—	0.4880	0.5490	—
HippoRAG 2	—	0.7170	0.6700	—
2WikiMultihopQA (closed-book 0.1862, gold 0.4685)				
Dense	0.2052	0.2082	0.2132	0.2242
RRF (BM25+Dense)+CE	0.2523	0.2492	0.2392	0.2342
ColBERTv2+CE	0.2563	0.2472	0.2352	0.2312
HEMLOCK	0.2643	0.2653	0.2793	**0.2933**
IRCoT (frozen reader)	—	0.2202	0.2052	—
HippoRAG 2	—	0.3243	0.2893	—
MuSiQue (closed-book 0.0280, gold 0.4400)				
Dense	0.0930	0.0990	0.0960	0.1050
RRF (BM25+Dense)+CE	0.1180	0.1180	0.1190	0.1200
ColBERTv2+CE	0.1210	0.1300	0.1220	0.1310
HEMLOCK	0.1230	0.1350	0.1740	**0.2080**

Q FURTHER LIMITATIONS

Shared misses with LinearRAG. When HEMLOCK and LinearRAG both fail to retrieve every gold document on a query, they miss the same documents 89.2%, 89.3%, and 79.5% of the time, against 36.8%, 37.8%, and 45.4% under a null that holds each system's miss count fixed and redraws which titles it missed (20,000 simulations; Figure 7). Every observed rate lies far outside the null's 95% interval. Those queries are 6.7%, 7.4%, and 34.4% of each benchmark, and on MuSiQue they cover a third of it. The two systems share no seed retriever and no scorer, so part of the residual looks like a property of the corpus rather than of either design.

Canonicalisation. Token-blocked fuzzy clustering fails in both directions: it conflates distinct entities with similar surface forms, and it cannot merge synonyms sharing no token, so "monarch" and "Queen" remain separate entities however often they co-refer.

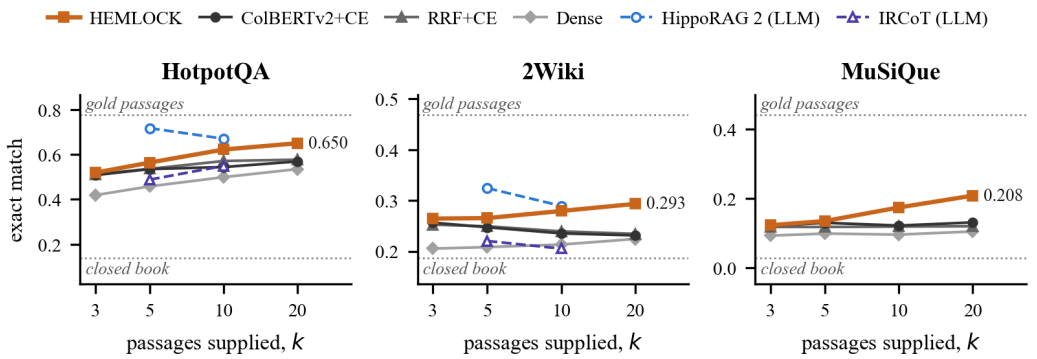


Figure 5: **Exact match under a frozen reader, unnormalised.** Raw exact match against k for every system; dotted lines are the closed-book and gold-context runs. HotpotQA is train.

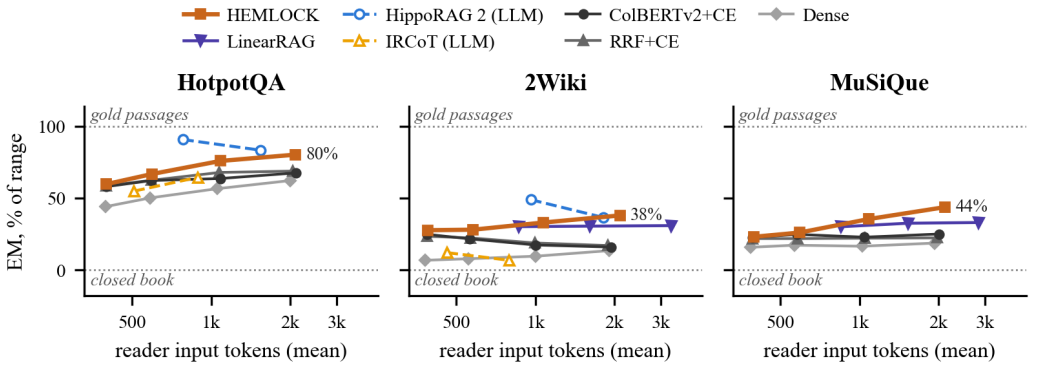


Figure 6: **Exact match against reader input tokens.** Exact match against reader input tokens, with every baseline. LinearRAG, at its published hyperparameters, appears only on 2WikiMultihopQA and MuSiQue because its HotpotQA reader run is on dev (Table 16). HotpotQA is train.

Reader dependence. Comparisons between retrievers hold the reader fixed; their magnitudes would change under a stronger reader, and their absolute values under a different question set.

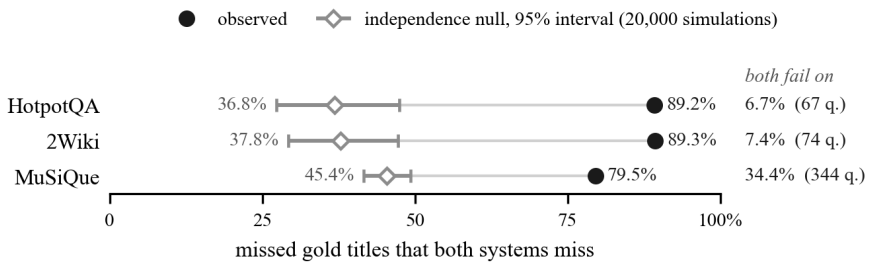


Figure 7: **Shared misses against an independence null.** On queries where both HEMLOCK (binary graph) and LinearRAG (λ tuned on the test questions) miss a gold document, the share of missed documents both systems miss. The null holds each system's miss count fixed and redraws which titles it missed. HotpotQA is dev.