

RECURSIVE DENSE RETRIEVAL

Anonymous Website Anonymous Code

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive retrieval comes in a few flavors: *single-hop* retrieval iteratively expands context around a single concept, whereas *multi-hop* retrieval focuses on *depth*, where it iteratively continues an evidence chain, and/or *width*, searching for complementary pieces of evidence. Although all scenarios require, at each retrieval step, decisions on (i) what to search for next and (ii) how to organize the evidence gathered along the way, both encoder training and inference strategies largely remain scenario-specific. In this work, we introduce RDR, a recursive dense retrieval framework that unifies single- and multi-hop retrieval. At the core of RDR is an instruction-conditioned encoder trained to map a retrieval goal and observed artifacts into a dense next-step query. Repeated encoding and index lookups build a retrieval tree, and we produce readouts (i.e., evidence rankings) by aggregating evidence scores across the tree's nodes. Because the instruction is part of the embedded state, the same encoder can both refine single-hop queries and follow multi-hop evidence chains. We train an RDR-8B encoder on *Recursive Retrieval Worlds* (RRW), a 4.75 million-sample dataset compiled from existing retrieval data, contextual transformations, constructed worlds, and recovery supervision. RDR performs strongly on a wide range of single- and multi-hop evaluations. In multi-hop retrieval, with a four-level retrieval tree with a branching factor of three, RDR reaches 80.14 and 25.28 `Recall@5` on MuSiQue and BrowseComp+, respectively, outperforming state-of-the-art encoders and multi-hop retrievers on all four multi-hop benchmarks. In single-hop retrieval, RDR's tree-based readout improves the encoder's five-benchmark mean by 2.5 points, most on conversational retrieval (70.79 `nDCG@10`) and on multi-aspect reasoning queries, where it nearly closes the gap to the strongest reasoning retriever. On average, RDR outperforms the strongest evaluated baselines in both task families, substantially on multi-hop retrieval (by 5.8 `Recall@5`) and by 3.6 `nDCG@10` on single-hop retrieval. We will publicly release RDR-8B, RRW data, and the evaluation code.

1 INTRODUCTION

Modern instruction-tuned embedding models support a wide variety of information retrieval tasks and use cases, from finding relevant passages to retrieving code and matching natural-language requests for tools (Su et al., 2023; Zhang et al., 2025). Finding an initial match, however, is often only the beginning: an observed document may reveal a missing fact, a conversation may change the information need, and a potentially relevant artifact may reveal nuances that the next search should distinguish. Often, a retriever performs individual lookups within an agentic loop, where a language model interprets retrieved artifacts and decides what to seek next (Asai et al., 2024; Wang et al., 2025; Liu et al., 2026). We argue that this is suboptimal in both efficiency (an additional generative LLM is required to formulate queries at each step) and effectiveness, because it fundamentally separates evidence retrieval from the decision of how to update the search.

In this work, we instead ask the following question: *Can a retriever itself learn to update its search based on what it observes?* Such an update ability offers a unifying framework across different retrieval objectives: (i) a single-hop search (Yu et al., 2021; Li et al., 2023; Albishre, 2026) can use feedback to improve relevance, (ii) a multi-hop question requires continuing an evidence chain (Xiong et al., 2021; Zhao et al., 2021; Erker et al., 2026), and (iii) a broad query requires collecting complementary evidence (Carbonell & Goldstein, 1998; Min et al., 2021; Ju et al., 2026). Each of

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.

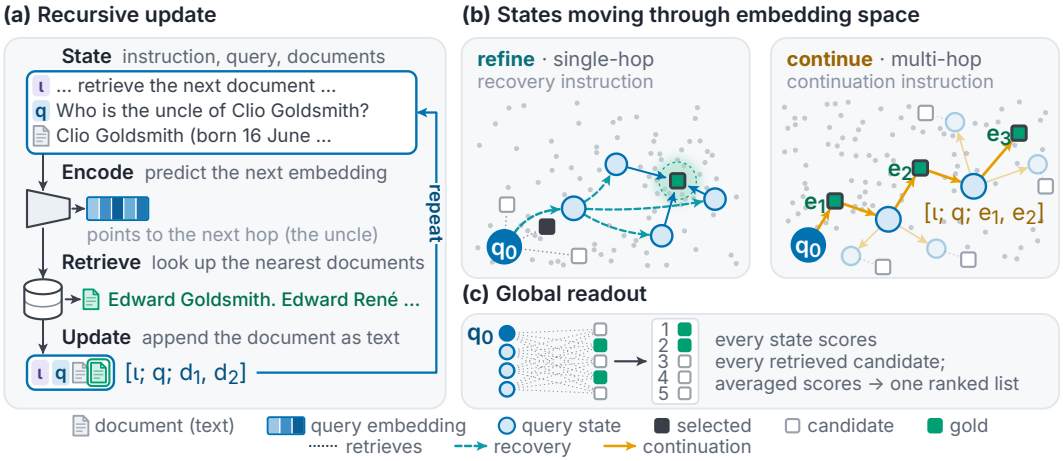


Figure 1: **RDR: unifying dense retrieval with a single learned operator that recursively composes contextual states.** **(a)** A state consists of an instruction, a query, and observed artifacts as text (two-hop illustration): the encoder predicts the next query embedding, which retrieves the second evidence document, which gets appended to the state. **(b)** Repeating the update moves the state through the embedding space (schematic): the *recovery* instruction refines a single-hop query, whereas the *continuation* instruction follows an evidence chain in multi-hop scenarios. (c) We score every retrieved candidate across all states and produce the global readout (i.e., ranking) by averaging the state-level scores (Equation 1); documents scored highly by several states rank higher.

these lines of work develops a scenario-specific a) encoder training, b) search/query update strategy, and c) readout (i.e., artifact ranking) approach, tailored and evaluated for one task family (Appendix K). Conversely, none has demonstrated the ability to generalize across heterogeneous retrieval scenarios or across artifact types, from web pages and dialogues to code, tool descriptions, and tables.

In contrast, as illustrated in Figure 1, we formulate the contextual update as a learned operator that, coupled with a search history and readout strategy, aims to generalize across retrieval scenarios, and that does so with a fixed document index (i.e., no document re-embedding) and without any text generation (e.g., for intermediate queries). Our RDR framework thus unifies recursive dense retrieval scenarios. At its core is a single instruction-conditioned encoder (i.e., operator) that learns to transform a retrieval goal and observed artifacts into a dense query embedding for the next search step. Repeated encoding and index lookup build a retrieval tree; a level-balanced global readout combines evidence scored across the tree's states (Figure 1). Our key finding concerns how search and readout combine: aggregating scores for each candidate across all composed states outperforms scoring it only by the path that selected it; this holds for both single-hop queries, which the operator iteratively refines, and multi-hop evidence chains. To our knowledge, RDR is the first retriever to read out a tree of learned, sequentially composed query states *globally*. Prior score aggregation efforts fuse parallel views that no learned update connects, and score each document only by its own path.

To generalize this capability across heterogeneous tasks, we build *Recursive Retrieval Worlds* (RRW): 4.75 million training samples in one instruction–query–observation–target format. Beyond bundling existing single-hop retrieval, multi-hop QA, and fact-checking data, we construct 1.7 million samples: we create Doc-ReasonIR, Chat-ReasonIR, and Doc-Promptriever by turning isolated retrieval tasks into document- and dialog-conditioned transitions; synthetic company worlds and persona-seeded web worlds provide complementary evidence and constrained-chain tasks; and recovery supervision augments existing retrieval data, including code, SQL, tool-call, and chat retrieval, with misleading or out-of-context observations. As a result, 43% of the samples include the observed documents in the context, the kind of state recursive search produces at test time (Figure 3).

Our RDR-8B model (trained on RRW), coupled with global readout, (1) exceeds the `Recall@5` of the multi-hop specialist model GRITHopper Erker et al. (2026) on all four multi-hop suites, and (2) improves its own single-step retrieval score on all six single-hop tasks. Further, it achieves the highest five-suite mean among the evaluated models (59.34 versus 55.79 for the backbone embedder), outperforms the conversational specialist ChatRetriever Mao et al. (2024) on the conversational retrieval benchmark, and nearly matches the performance of the reasoning specialist BGE-Reasoner

Chen et al. (2026a) on aspect-aware BRIGHT-Pro Zhao et al. (2026). Comparison against the backbone (Qwen3-Embedding-8B) run on the same search trees shows that this cross-task generalization stems from the learned contextual update (not from more inference computation).

Our work addresses the following research questions: **RQ1 (multi-hop):** Can the same shared contextual operator support both evidence-chain retrieval and broader evidence coverage in multi-hop tasks? **RQ2 (single-hop):** Can the same framework—model and recursive search—also improve single-hop retrieval across general, conversational, reasoning-intensive, code-, and tool-based tasks? **RQ3 (test-time scaling):** How do individually supervised contextual updates compose as depth H and branching b increase, and what accuracy–computation–latency trade-offs do they offer? **RQ4 (search and readout):** Which combinations of search and readout strategies let one contextual operator serve both single-hop and multi-hop retrieval?

We introduce a unifying framework for recursive dense retrieval, with the following contributions:

① **One contextual operator across retrieval scenarios and artifacts.** RDR-8B learns instruction-conditioned updates (from 4.75 million RRW samples) and uses them to refine single-hop queries over documents, dialogues, code, and tools, as well as to follow evidence chains and collect complementary evidence for multi-hop queries.

② **A global readout over sequentially composed states.** RDR ranks every retained candidate by aggregating the scores it receives across all learned states of its retrieval tree. Coupling different search strategies with our global readout shows that our readout drives performance gains: we benefit from more states producing relevance predictions.

③ **Cross-task evaluation.** Across a broad range of multi-hop and single-hop experiments, we connect (i) our learned contextual updates (i.e., the operator) with (ii) our global readout strategy, profiling depth–breadth trade-offs in recursive retrieval.

2 Related Work

Three key aspects span the landscape of work on recursive retrieval:

Test-time refinement. Pseudo-relevance feedback updates a single-hop query from retrieved passages, re-encoding them jointly into one query vector (ANCE-PRF; Yu et al., 2021), combining embeddings (Li et al., 2023), or refining a frozen encoder's query vector sequentially and interpolating it with the original query (DC-PRF; Albishre, 2026). RDR instead trains the contextual update model itself and aggregates relevance scores across all search states.

Decomposition-free multi-hop retrieval. MDR and BeamDR condition subsequent retrieval on observed passages, as do earlier multi-step retrievers (Das et al., 2019; Asai et al., 2020; Khattab et al., 2021), and GRITHopper extends this with decoder-based instruction tuning (Xiong et al., 2021; Zhao et al., 2021; Erker et al., 2026). They learn chained updates, but return realized paths: documents that are scored but not selected are discarded, and each returned document is scored only by its own chain.

Search and aggregation. Fusing ranked lists over one index is classical (Fox & Shaw, 1994; Cormack et al., 2009), and fusing query variations improves effectiveness (Benham & Culpepper, 2017). Chain of Retrieval (COR) decomposes a query paper into LLM-generated aspect queries, grows a tree by promoting retrieved papers to new queries, and fuses the lists post-order by reciprocal rank (Park et al., 2025); MMR, JPR, and CoveR select complementary or covering evidence (Carbonell & Goldstein, 1998; Min et al., 2021; Ju et al., 2026). None of these fuses produces states learned via an update conditioned on the observed artifacts.

RDR learns the contextual update of the first two lines of work, composes it recursively, and reads out the resulting sequence of states globally by their scores, so a candidate that several states score highly ranks high even if no path selected it. RDR's learned updates connect the aggregated states; the agreement among them thus reflects evidence gathered along the search, not just agreement between parallel views.

3 Recursive Dense Retrieval

Instruction-conditioned next-state prediction. Let $\mathcal{D}$ be a corpus with cached document vectors e_d. For an instruction ι, query q, and ordered observations O_s, we define the search state s, next-step query

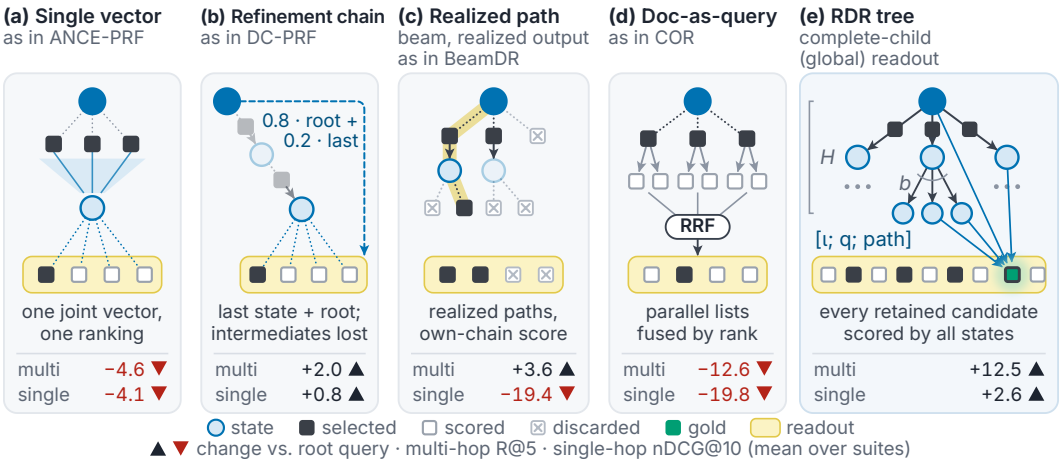


Figure 2: **Search and readout strategies (under the same encoder).** (a) Pseudo-relevance feedback encodes observations jointly into one query vector. (b) As in DC-PRF, a refinement chain reads out its last state, interpolated with the root. (c) Beam retrievers return realized paths, each document scored by its own chain. (d) Promoting retrieved documents to queries fuses parallel lists by rank. (e) RDR scores every retained candidate with all learned states. Numbers: change w.r.t. the initial query in multi-hop `Recall@5` and single-hop `nDCG@10` (Tables 3 and 24); (c) is BeamDR on a 5-wide beam.

vector z_s, and document scores $\ell_s(d)$ as $s = [\iota; q; \texttt{ctx}(O_s)]$, $z_s = f_\theta(s)$, and $\ell_s(d) = \cos(z_s, e_d)$, respectively, where `ctx` produces a text representation of observations O_s and f_θ denotes the encoder model (i.e., the contextual operator). A retrieved document grounds the query prediction and can become an observation for the next step, that is, the next call to the same encoder. We serialize documents, tables, code, and conversations as text (the `ctx` function). As is common in instruction-tuned retrievers, root instructions are dataset-specific (Appendix B.5). Document embeddings e_d remain fixed during the whole query-side search.

The two instruction types induce complementary contextual views. *Continuation* asks for still-missing evidence given the observations; *recovery* pushes to revise the query to match the initial goal better. Recovery supplies the primary update for relevance refinement; continuation supplies the update for evidence completion, including aspect coverage. Because the instruction is part of the state s, re-encoding the same observed artifacts O_s under continuation vs. recovery instruction types yields complementary retrieval views: a simple, parameter-free fusion of both can thus improve retrieval without changing the observed evidence or the traversal (Section 5.1; Appendix E.1). Both roles apply in both task families: on single-hop suites, trees driven by either instruction type improve over the root (Appendix G); on multi-hop suites, re-encoding a continuation tree's observations with the recovery instruction yields a complementary view that alone ranks candidates about as well as the tree's own states and, fused with them, ranks best (Table 20).

A retrieval tree with level-balanced aggregation RDR(H, b) starts with the empty-context root and evaluates up to H levels, counting the root as level 1, i.e., the deepest states have observed $H - 1$ documents. Each expandable state selects up to b documents, and each child appends one of those selected documents to its parent's observation path. We use RDR$(H = 4, b = 3)$ as a default configuration: $H = 4$ matches the (up to) four hops of MuSiQue and GRITHopper's evaluation (Erker et al., 2026), with at most $1 + 3 + 9 + 27 = 40$ primary query states and up to three ancestor documents at the last level. Our readout approach is agnostic to the search strategy: for each document d, we aggregate its relevances $p_s(d)$ across all states s. I.e., our readout r equally applies to chains (i.e., greedy search), beams, and full search trees. Let C_s be the K top-ranked documents for state s, let F_h denote all states at a level h, and let L be the number of levels for which $F_h \neq \emptyset$. We then compute the state-specific relevances $p_s(d)$ and the global readout $r(d)$ as follows:

$$p_s(d) = \frac{\mathbf{1}[d \in C_s] \exp(\ell_s(d)/\tau)}{\sum_{u \in C_s} \exp(\ell_s(u)/\tau)}, \qquad r(d) = \frac{1}{L} \sum_h \frac{1}{|F_h|} \sum_{s \in F_h} p_s(d). \tag{1}$$

Unless specified otherwise, we set $\tau = 0.05$, and retrieve top $K = |C_s| = 200$ documents per state.

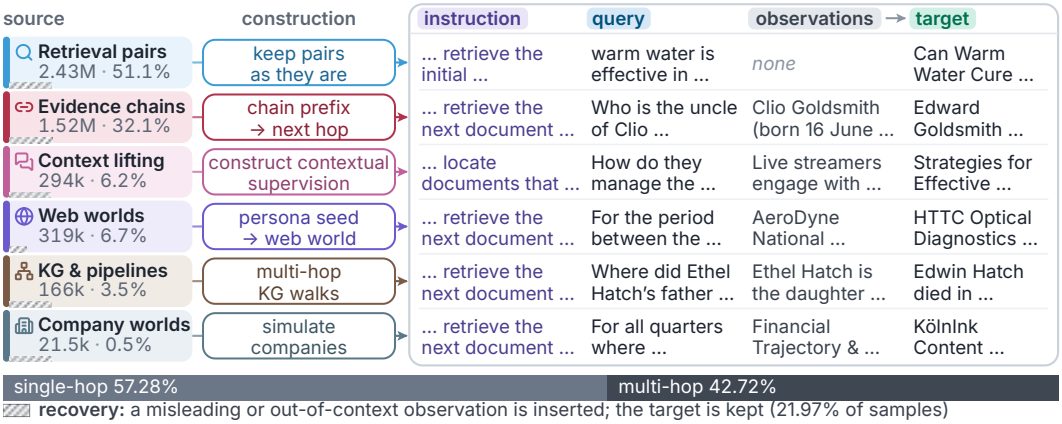


Figure 3: **Recursive Retrieval Worlds: many sources, one training tuple.** Retrieval pairs, evidence chains, contextualized tasks, and constructed web, knowledge-graph, and company worlds all become the instruction–query–observation–target tuple used by RDR's update; recovery samples (21.97%) insert a misleading observation and keep the target.

Our readout is an instance of a CombSUM aggregation (Fox & Shaw, 1994): within each search level (i.e., depth of the search tree), we average the softmax-normalized top-K scores with equal weight. Each state contributes its sparse score distribution p_s over its top-K candidates, covering many documents beyond those selected to extend the tree (i.e., $K = 200 \gg b = 3$). As a result, a document ranks high when it appears in C_s across many states at different levels, with later-level states re-scoring documents that earlier-level states retrieved. Branching induces alternate observation paths. Level-balancing lets these paths contribute without overweighting larger frontiers and keeps the root's goal anchored. Appendix B specifies output processing; Figure 2 contrasts our readout against prior approaches. The anonymous website walks through Equation 1 on a toy tree and animates each search–readout combination.

4 Recursive Retrieval Worlds

4.1 Training corpus

RRW contains 4.75 million training samples. Each sample pairs an instruction and query with a target artifact and, where available, observed context (Section 3). Contextual samples include evidence-chain prefixes, document- and dialogue-conditioned queries, constructed information worlds, and recovery examples. Standard query–target samples complement this supervision to retain single-hop retrieval performance. The dataset contains 2.03 million multi-hop samples (42.72%) and the remaining 57.28% are single-hop samples, including contextual tasks. Across both groups, 1.04 million samples (21.97%) provide recovery supervision.

4.2 Contextualization and information-world construction

We combine existing retrieval datasets with contextual transformations and synthetic information worlds. **Existing evidence chains** come from multi-hop QA—MuSiQue (Trivedi et al., 2022), HotpotQA (Yang et al., 2018), and 2WikiMultiHopQA (Ho et al., 2020)—and fact-checking, including HoVer (Jiang et al., 2020) and EX-FEVER Ma et al. (2024): these are the five training datasets of GRITHopper (Erker et al., 2026), with chains of up to four documents (MDR and BeamDR train on two-hop chains). BrowseComp+ Chen et al. (2026b) and FanOutQA Zhu et al. (2024), in contrast, need evidence from a median of six different source documents per question (Table 4). Each successful prefix provides observations to predict the next artifact. **Context lifting** turns reasoning and instruction supervision into document- and dialogue-conditioned transitions (Doc-ReasonIR, Chat-ReasonIR, and Doc-Promptriever; Shao et al., 2025; Weller et al., 2025), whereas **constructed** company and web **worlds** utilize personas as seeds Ge et al. (2025) to supply complementary-evidence and constrained-chain supervision beyond four documents: 96k of their 341k samples target the fifth to eighteenth document of a chain or an evidence set (see Figure 3 and Appendix C for more details).

Our data viewer shows 50 samples of each of our splits (e.g., Doc-Promptriever, Chat-ReasonIR) and the prompts and intermediate outputs of our data construction.

Recovery supervision contextualizes any sample, including plain single-hop query–target pairs: it inserts a misleading observation, a hard negative, or an out-of-context document while keeping the target, so the encoder learns to return to the original need from a misleading context. Samples that already carry context, such as dialogues and evidence-chain prefixes, also receive the observation. Recovery covers 1.04 million samples (21.97%), including recovery versions of 63 existing single-hop datasets for code, SQL, tool-call, and chat retrieval (hatched in Figure 3). At test time, the tree's retrieved documents replace these observations.

4.3 Training

For state s, target d^+, and hard/in-batch alternatives $\mathcal{N}_s$, we compute the following loss:

$$\mathcal{L}(s, d^+) = -\log \frac{\exp(\cos(f_\theta(s), e_{d^+})/\tau_{\text{train}})}{\sum_{d \in \{d^+\} \cup \mathcal{N}_s} \exp(\cos(f_\theta(s), e_d)/\tau_{\text{train}})}. \tag{2}$$

Targets d^+ are (pre-)indexed, with embeddings e_{d^+}. We tokenize training queries and passages to at most 4,096 and 2,048 tokens, respectively, truncating when needed. We fully fine-tune Qwen3-Embedding-8B (Zhang et al., 2025) with $\tau_{\text{train}} = 0.02$ over one hard negative per sample and in-batch negatives shared across devices, a learning rate of 10^{-5} with a warmup–stable–decay schedule, and an effective batch size of 2,048 on 32 A100 GPUs, using gradient caching (Gao et al., 2021). We shuffle the mixture without any task-based loss weighting. We train RDR-8B for one epoch on 4.75 million samples (further training details are in Appendix C). For constructed samples, we apply source-specific quality checks, de-duplication, and negative validation (details in Appendix C); these recipes will accompany the data release.

4.4 Evaluation Setup

Benchmarks. BRIGHT (Su et al., 2025) tests reasoning-intensive retrieval; NanoBEIR (Zeta Alpha, 2024; Thakur et al., 2021) general retrieval, CoIR (Li et al., 2025a) code retrieval, ToolRet (Shi et al., 2025) tool selection, and TopiOCQA (Adlakha et al., 2022) conversational retrieval. These use `nDCG@10`; BRIGHT-Pro (Zhao et al., 2026) uses α-`nDCG@10`. MuSiQue (Trivedi et al., 2022) and BrowseComp+ (Wei et al., 2025; Chen et al., 2026b) assess chained-evidence retrieval; FanOutQA (Zhu et al., 2024) and FRAMES (Krishna et al., 2025) assess evidence-group coverage. We report `Recall@5/@10` (labels in Appendix B).

Baselines. We test how close RDR comes to specialist approaches in respective tasks, while generalizing across all retrieval task families. Baselines thus comprise general-purpose embedders: Qwen3-Embedding-8B as RDR's initialization, and Octen (Octen Team, 2026), as well as specialist models for one task family: ReasonIR (Shao et al., 2025) and BGE-Reasoner (Chen et al., 2026a) for reasoning-intensive retrieval, ChatRetriever (Mao et al., 2024) for conversational retrieval, CoveR (Ju et al., 2026) for coverage, and GRITHopper-7B, a recursively applied bidirectional encoder, for multi-hop retrieval; for each we adopt their respective recommended inference procedure. We provide the results of our tree-based readout on top of GRITHopper's encoder in Appendix C.3. The initialization runs under the same search as RDR over its own document index. For the five `nDCG@10` suites, the primary tree uses the recovery instruction at every context-bearing level; evidence completion and BRIGHT-Pro use continuation, with one wording for all suites (Appendix B.5). Recovery fusion (explained in Appendix E.1) and Assembly (explained in Appendix I.1) are optional extensions and are reported separately from the primary tree.

5 Experiments

5.1 RQ1: Multi-hop retrieval from depth to width

Table 1 shows effectiveness in retrieving chained evidence (depth-based multi-hop) and covering complementary source groups (width-based multi-hop); the latter requires selecting evidence across paths rather than extending a single evidence chain.

RDR-8B with RDR(4, 3) improves over single-step retrieval (i.e., root query) by 21.63, 8.41, 9.15, and 10.63 `Recall@5` points on MuSiQue, BrowseComp+, FanOutQA, and FRAMES, respectively. It significantly exceeds GRITHopper on all four suites (by 1.6–12.3 `Recall@5` points; with all 95%

Table 1: Multi-hop retrieval, Recall@5/@10. Assembly selects ten documents from the tree's own pool. Recovery: an optional z-agreement aggregation (not a change to the default aggregator from Eq. 1) (see Appendix E.1.1). Bold: best without the optional extensions.

	Chained evidence				Coverage			
	MuSiQue		BrowseComp+		FanOutQA		FRAMES	
Model / configuration	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
Qwen3-Embedding-8B, single-step	63.16	72.08	14.70	19.98	26.79	40.43	67.80	77.99
same raw tree, RDR(4, 3)	62.68	75.51	16.91	25.40	27.65	40.04	68.25	76.62
ReasonIR-8B, single-step	53.43	63.77	9.86	14.20	28.40	42.70	63.05	74.30
Octen-Embedding-8B, single-step	62.20	71.20	14.66	20.77	26.28	37.59	64.75	76.28
BGE-Reasoner-8B, single-step	44.95	56.61	12.12	17.05	28.65	42.44	58.55	69.53
CoveR (0.1B), single-step	50.32	58.15	7.87	10.98	20.59	29.26	54.64	64.87
GRITHopper-7B, recursive	78.55	86.87	12.98	18.04	31.96	43.83	74.08	83.51
RDR(1, 0) (single-step)	58.51	65.99	16.87	23.10	28.63	42.72	66.77	76.83
RDR(4, 3)	**80.14**	**87.41**	**25.28**	**33.38**	**37.78**	**57.07**	**77.41**	**86.28**
+ recovery (z-agreement)	81.45	87.56	25.28	35.02	40.66	60.26	77.89	87.21
+ Assembly	79.62	82.86	23.63	30.23	49.29	63.52	79.20	83.43

Table 2: Single-hop retrieval, nDCG@10 (BRIGHT-Pro: α-nDCG@10). RDR(4, 3) uses continuation for BRIGHT-Pro and recovery for the other suites. BRIGHT includes all twelve domains. Avg. weights the other five suites equally, excluding BRIGHT-Pro's different metric. Shallow-breadth controls retain their single-observation configuration and paired root baseline (Appendix D).

Model/configuration	BRIGHT-Pro	BRIGHT	NanoBEIR	ToolRet	CoIR	TopiOCQA	Avg.
Qwen3-Embedding-8B	43.76	23.17	**71.74**	44.72	**79.93**	59.40	55.79
ReasonIR-8B	34.98	24.18	59.81	40.59	56.81	42.61	44.80
Octen-Embedding-8B	35.10	20.45	69.35	43.96	78.88	55.02	53.53
BGE-Reasoner-8B	**61.82**	**36.97**	60.90	42.26	65.32	41.85	49.46
ChatRetriever-7B	8.72	10.73	56.12	34.06	49.90	69.02	43.97
RDR(1, 0) (single-step, no context)	55.92	29.02	66.55	46.45	78.77	63.37	56.83
RDR(4, 3) (shared tree)	**61.13**	**30.84**	**68.13**	48.04	**78.88**	**70.79**	**59.34**
Ablation: shallow breadth, same operator, $H = 2$							
RDR(2, 2)	58.85	29.90	67.49	47.62	78.79	64.00	57.56
RDR(2, 8)	59.23	30.16	67.96	**48.08**	78.84	66.49	58.31
RDR(2, 16)	58.86	30.35	68.01	47.99	78.84	66.73	58.38

confidence intervals above zero). Under the same search tree, the initialization (Qwen3-Embedding-8B) moves marginally (−0.48, +2.21, +0.86, and +0.45). This proves that the RDR's gains stem from its contextual update operator (RDR-8B) and the readout strategy, and not merely from more search (i.e., test-time) compute. The anonymous website compares models' rankings for example queries.

Re-encoding the same non-root contexts in a recovery view (i.e., with a recovery instead of continuation instruction), which marks the last observation as incorrect, leaves the primary traversal unchanged. Optional z-agreement aggregation of the two views improves performance by an average of 1.17 Recall@5 points, suggesting that multi-hop retrieval may benefit from complementary instruction-conditioned views (i.e., encodings) of the same observations. GRITHopper, trained only to continue evidence chains, cannot recover in this way. Appendix E.1 specifies actual recovery eligibility and the alternative level-only readout.

An optional Assembly, a set-conditioned selection of ten documents from the tree's candidate pool (Appendix I.1), targets redundant evidence across branches: it adds 11.51 Recall@5 on FanOutQA and 1.79 on FRAMES but costs 0.52 on MuSiQue and 1.65 on BrowseComp+ (more in Appendix I).

Takeaway — RQ1

Our RDR decomposition-free encoder (with the global readout strategy) outperforms a multi-hop specialist GRITHopper on all multi-hop suites. Under the same search tree, RDR's backbone (Qwen3-Embedding-8B) yields marginal gains, rendering RDR-8B highly effective.

5.2 RQ2: Does recursion also improve single-hop retrieval?

Table 2 summarizes the single-hop retrieval results. RDR improves over its own root (i.e., single-step retrieval with RDR-8B) on all five primary suites and has the highest mean among the evaluated models.

Some specialist models have an advantage on their specialized tasks (e.g., RDR trails BGE-Reasoner by 6.1 on BRIGHT), but fall far below RDR elsewhere (five-suite means 49.5 for BGE-Reasoner and 44.0 for ChatRetriever vs. RDR's 59.3). In contrast, RDR exceeds the conversational specialist and nearly matches BGE-Reasoner on the multi-aspect BRIGHT-Pro benchmark. Contextual states help the most when the information need unfolds over a dialogue or exhibits several aspects: our global-tree readout brings 7.4 points over single-step RDR retrieval on the conversational benchmark (TopiOCQA), slightly outperforming the specialist ChatRetriever's 69.02 (Appendix B.2); the tree-readout also brings 5.21 performance points on BRIGHT-Pro. On BRIGHT, NanoBEIR, and ToolRet, the gains over the root are smaller but consistent (1.6–1.8 points); they are only marginal on CoIR (+0.11). The cross-task utility stems from recursive inference: on NanoBEIR, RDR's single-step retrieval trails the backbone Qwen by 5.1 points, but the tree recovers 1.6 points.

Multi-aspect reasoning retrieval. BRIGHT-Pro's weighted-aspect evaluation (Zhao et al., 2026) credits covering several aspects of a reasoning-intensive query. Here, the continuation tree gains 5.2 points over its root, the largest test-time gain among the reasoning suites, and reaches 61.1, nearly closing its root's 5.9-point gap to BGE-Reasoner (61.8) (see the Appendix G.1).

Takeaway — RQ2

Recursion also helps single-hop retrieval, but only with a trained contextual update: the same tree adds 2.50 points to RDR's five-suite mean and costs the initialization 1.35. The gains are largest on conversational and multi-aspect reasoning retrieval and smaller but positive on the other suites.

5.3 RQ3: Generalization over test-time depth and breadth

Our training supervision consists only of individual state updates; we compose them by width and depth at inference to construct the search tree. We next analyze the depth-breadth tradeoffs under a fixed number of states (i.e., tree nodes).

Depth and breadth. RDR$(2, m)$ denotes a root followed by m independently observed candidates, read out the same way; it requires $1 + m$ encodings in two dependent rounds (compared to four rounds in RDR$(4, 3)$). At a maximum of 40 states, RDR$(4, 3)$ exceeds RDR$(2, 39)$ on all multi-hop suites, by 5.9–13.2 `Recall@5` points (Table 15). On single-hop tasks, searching over shallow trees improves the respective root performance by 0.77, 1.51, and 1.59 mean points for $m = 2, 8, 16$, respectively (Figure 4). On TopiOCQA, the recovery tree gains 7.4 points at its first contextual level and retains this gain through to $H = 4$ (Table 6).

Quality–cost trade-off. On MuSiQue, RDR$(3, 3)$ reaches 79.9 `Recall@5` at 0.36 s/query, versus 80.1 at 1.02 s/query for RDR$(4, 3)$ (Table 18). Shallow trees batch independent updates in two rounds, which offers a lower-cost operation in single-hop retrieval. With the global readout, a single *chain* of four encodings reaches 80.7 `Recall@5` on MuSiQue at 0.15 s/query and 53.9 mean `Recall@5` over the four suites, which is 1.2 points below the full-tree performance (Table 3b); while the chain is more efficient, the tree is a more robust all-rounder.

Takeaway — RQ3

At a fixed budget of 40 states, depth beats breadth on multi-hop retrieval by 5.9–13.2 `Recall@5`. But breadth helps even with shallow trees (depth 2) in single-hop retrieval; RDR$(3, 3)$ retains 99.7% of RDR$(4, 3)$ `Recall@5` performance (on MuSiQue), at about a third of the latency.

5.4 RQ4: Search and readout across retrieval tasks

Search determines which states form, whereas readout determines how states' scores turn into a document ranking. We now cross the design choices for both.

Table 3a shows the inference forms of prior retrievers with our RDR-8B encoder (i.e., without their training). RDR stays above all of them. On multi-hop retrieval, DC-PRF's root interpolation and a 5-wide beam (as in BeamDR) gain 2.0–3.6 points over the root alone, while chain-ordered outputs help most on FanOutQA, where whole chains cover more distinct evidence. On single-hop retrieval, DC-PRF's interpolation gains 0.8 points over the root but stays 1.8 points below RDR; the joint feedback inference loses 4.1 points. The beam and document-expansion outputs are built to return chains or paper sets, not to rank one information need; they thus fall 19–23 points below the root (Appendix G).

Table 3: Search and readout under one encoder (RDR-8B, same index). Multi-hop: mean Recall@5 over four suites; single-hop: mean nDCG@10 over the five suites of Table 2 (recovery instruction; CoIR on a 17,901-query sample). (a) Inference forms of prior work with RDR's encoder, evaluated as ranked lists; DC-PRF at its root weight $\lambda = 0.8$. (b) RDR's states on each search under three readouts; single-hop path products are completed with the root ranking. *Built to return chains or paper sets; values in Table 24. Bold: best in (a). Per-suite results: Appendices F, G, and H.

	Search / readout	Multi-hop	Single-hop
(a) Configurations			
Root query	—	42.70	56.78
as in ANCE-PRF	joint feedback / single vector	38.15	52.71
as in DC-PRF	chain / last state, root-interpolated	44.68	57.57
as in BeamDR	beam (W=5 / 13) / probability chain	46.30 / 40.63	n/a*
as in MDR	beam (W=5 / 13) / sequence sum	44.72 / 38.98	n/a*
as in COR	document-as-query tree / rank fusion	30.08	n/a*
RDR	learned tree / global readout	**55.15**	**59.34**
(b) Readout versus search			
	chain / terminal state	32.19	54.31
	chain / global readout	53.91	57.95
	beam (W=13) / path product	42.77	56.68
	beam (W=13) / global readout	53.88	59.60
	learned tree / path product	52.69	57.65
	document-as-query tree / global readout	40.41	55.03

RDR improves over the root in both task families (+12.5 Recall@5, +2.6 nDCG@10), and Table 3b reveals why: the readout is critical. With the global readout, chain, beam, and tree reach 53.9–55.2 mean Recall@5 and 58.0–59.6 nDCG@10: the search matters far less than the readout; branching adds significant Recall@5 gains over a single chain on BrowseComp+ (+2.14) and FRAMES (+3.32) but not on MuSiQue or FanOutQA (Appendix H). With RDR's encoder, every learned search with the global readout stays above the strongest baseline means in Tables 1 and 2 (49.4 Recall@5 and 55.8 nDCG@10). The path product, in contrast, gains 10 points over the root and almost nothing over the beam, whose top ranks are all root document picks. On the same tree, and at no extra cost, the global readout exceeds the path product by a mean of 2.5 Recall@5 points on multi-hop and 1.7 nDCG@10 points on single-hop retrieval. These gains stem from re-scoring selected documents with all states (Appendices F and G); documents supported by more states are more often relevant at every root-score level (Figure 5).

States must be learned: using document-as-query expansion instead, with the same readout, results in a massive 14.7 pp drop (App H); the same tree also costs the backbone 1.3 nDCG@10 points (Sec 5.2).

Takeaway — RQ4

RDR outperforms inference forms of prior retrievers: the readout is key, not the search strategy. Scoring every candidate with all learned states outperforms path-based scoring.

6 Ablations and further analysis

Two controls isolate RDR's components: backbone under the same tree (learned update versus inference computation; Sections 5.1–5.2) and a fixed encoder (readout versus search; Section 5.4). At matched shallow budgets, RDR improves over its own root in all eight single-hop suite/budget cells and the backbone in none (Appendix C, Table 11). RDR's candidate pools also carry over to LLM re-ranking: a 27B listwise judge re-ranking of RDR's top-50 exceeds the same judge on the single-query pool by 2.3–10.2 Recall@5 on all four multi-hop suites (Appendix J).

7 Conclusion

RDR composes learned, instruction-conditioned state updates without decomposition or a generator and reads out the resulting tree globally. One operator refines single-hop queries over documents, dialogues, code, and tools, follows evidence chains, and collects complementary evidence; its global readout outperforms path scoring and inference of prior retrievers. Under the same search, the backbone model gains little. In sum, RDR offers a unified framework for recursive dense retrieval.

Reproducibility statement

Appendix B specifies evaluation, scoring, and input-processing settings; Appendix C describes the training corpus and optimization. The code repository contains the `rdr` package, including evaluation code, input templates, and one configuration per group of tables, documented on the anonymous website. Upon publication, we will release the model weights, construction recipes, and source-level data documentation. We will distribute RRW components in the forms permitted by their upstream terms.

AI use statement

LLMs generated and transformed synthetic training examples. LLM assistants also supported implementation, debugging, log analysis, result aggregation, and manuscript editing. The authors are responsible for the research questions, method, experimental design, and claims, and verify reported results against their evaluation outputs.

References

Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy. TopiOCQA: Open-domain conversational question answering with topic switching. *Transactions of the Association for Computational Linguistics*, 10:468–483, 2022. doi: 10.1162/tacl_a_00471. URL https://aclanthology.org/2022.tacl-1.27/. 6

Khaled Albishre. Anchored dense-chain pseudo-relevance feedback: sequential state refinement for neural retrieval. *Journal of King Saud University – Computer and Information Sciences*, 2026. doi: 10.1007/s44443-026-00604-x. 1, 3, 28, 33

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. Learning to retrieve reasoning paths over Wikipedia graph for question answering. In *International Conference on Learning Representations*, 2020. 3, 33

Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 3650–3675, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.225. URL https://aclanthology.org/2023.findings-acl.225/. 33

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.11511. 1, 34

Rodger Benham and J. Shane Culpepper. Risk-reward trade-offs in rank fusion. In *Proceedings of the 22nd Australasian Document Computing Symposium*, 2017. doi: 10.1145/3166072.3166084. 3

Jaime G. Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pp. 335–336, New York, NY, USA, 1998. Association for Computing Machinery. doi: 10.1145/290941.291025. URL https://doi.org/10.1145/290941.291025. 1, 3, 33

Hung-Ting Chen, Xiang Liu, Shauli Ravfogel, and Eunsol Choi. Beyond single embeddings: Capturing diverse targets with multi-query retrieval. *arXiv preprint arXiv:2511.02770*, 2025. URL https://arxiv.org/abs/2511.02770. 33

Jianlyu Chen, Junwei Lan, Chaofan Li, Defu Lian, and Zheng Liu. ReasonEmbed: Enhanced text embeddings for reasoning-intensive document retrieval. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1203–1221, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.54. URL https://aclanthology.org/2026.acl-long.54/. 3, 6

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Hosna Oyarhoseini, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A fair and disentangled evaluation benchmark for deep search agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22349–22370, San Diego, California, United States, July 2026b. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1023. URL https://aclanthology.org/2026.acl-long.1023/. [5, 6, 16]

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759. Association for Computing Machinery, 2009. doi: 10.1145/1571941.1572114. URL https://doi.org/10.1145/1571941.1572114. [3, 34]

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*, 2019. [3, 33]

Adnan El Assadi, Niklas Muennighoff, and Jinhyuk Lee. The embedder's dilemma: LLMs are better, but at what cost? In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=3vnhrqmgb5. [31, 32]

Justus-Jonas Erker, Florian Mai, Nils Reimers, Gerasimos Spanakis, and Iryna Gurevych. Triple-encoders: Representations that fire together, wire together. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5317–5332, 2024. doi: 10.18653/v1/2024.acl-long.290. URL https://aclanthology.org/2024.acl-long.290/. [34]

Justus-Jonas Erker, Nils Reimers, and Iryna Gurevych. GRITHopper: Decomposition-free multi-hop dense retrieval. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 77–94, 2026. doi: 10.18653/v1/2026.eacl-long.5. URL https://aclanthology.org/2026.eacl-long.5/. [1, 2, 3, 4, 5, 33]

Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pp. 243–252. National Institute of Standards and Technology, 1994. URL https://trec.nist.gov/pubs/trec2/t2_proceedings.html. [3, 5, 34]

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP*, pp. 316–321, 2021. doi: 10.18653/v1/2021.repl4nlp-1.31. URL https://aclanthology.org/2021.repl4nlp-1.31/. [6]

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas, 2025. URL https://arxiv.org/abs/2406.20094. [5]

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From RAG to memory: Non-parametric continual learning for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 21497–21515, 2025. [34]

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 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/. [5, 18]

Yichen Jiang, Shikha Bordia, Zheng Zhong, Charles Dognin, Maneesh Singh, and Mohit Bansal. HoVer: A dataset for many-hop fact extraction and claim verification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3441–3460, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.309. URL https://aclanthology.org/2020.findings-emnlp.309/. [5, 18]

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan O. Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2503.09516. [34]

Jia-Huei Ju, Eugene Yang, Trevor Adriaanse, Suzan Verberne, and Andrew Yates. Search for coverage: Learning coverage-aware retrieval with augmented sub-question answerability. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 723–733, New York, NY, USA, 2026. Association for Computing Machinery. doi: 10.1145/3805712.3809752. URL https://doi.org/10.1145/3805712.3809752. [1, 3, 6, 33]

Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. In *Advances in Neural Information Processing Systems*, 2021. [3, 33]

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4745–4759, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.243. URL https://aclanthology.org/2025.naacl-long.243/. [6]

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 17606–17619, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.861. URL https://aclanthology.org/2025.acl-long.861/. [33]

Hang Li, Ahmed Mourad, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Pseudo relevance feedback with deep language models and dense retrievers: Successes and pitfalls. *ACM Transactions on Information Systems*, 41(3), 2023. doi: 10.1145/3570724. URL https://doi.org/10.1145/3570724. [1, 3, 33]

Xiangyang Li, Kuicai Dong, Yi Quan Lee, Wei Xia, Hao Zhang, Xinyi Dai, Yasheng Wang, and Ruiming Tang. CoIR: A comprehensive benchmark for code information retrieval models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22074–22091, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1072. URL https://aclanthology.org/2025.acl-long.1072/. [6]

Zhonghao Li, Kunpeng Zhang, Jinghuai Ou, Shuliang Liu, and Xuming Hu. TreeHop: Generate and filter next query embeddings efficiently for multi-hop question answering. *arXiv preprint arXiv:2504.20114*, 2025b. [33]

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-R: Learning to retrieve for agentic search. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15987–16005, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.785. URL https://aclanthology.org/2026.findings-acl.785/. [1]

Huanhuan Ma, Weizhi Xu, Yifan Wei, Liuji Chen, Liang Wang, Qiang Liu, Shu Wu, and Liang Wang. EX-FEVER: A dataset for multi-hop explainable fact verification. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9340–9353, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.556. URL https://aclanthology.org/2024.findings-acl.556/. [5, 18]

Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. Adaptive re-ranking with a corpus graph. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022. [34]

Kelong Mao, Chenlong Deng, Haonan Chen, Fengran Mo, Zheng Liu, Tetsuya Sakai, and Zhicheng Dou. ChatRetriever: Adapting large language models for generalized and robust conversational dense retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1227–1240, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.71. URL https://aclanthology.org/2024.emnlp-main.71/. [2, 6]

Sewon Min, Kenton Lee, Ming-Wei Chang, Kristina Toutanova, and Hannaneh Hajishirzi. Joint passage ranking for diverse multi-answer retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6997–7008, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.560. URL https://aclanthology.org/2021.emnlp-main.560/. [1, 3, 33]

John X. Morris and Alexander M. Rush. Contextual document embeddings. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.02525. [33]

Octen Team. Octen series: Optimizing embedding models to #1 on RTEB leaderboard, 2026. URL https://octen-team.github.io/octen_blog/posts/octen-rteb-first-place/. [6]

Sangwoo Park, Jinheon Baek, Soyeong Jeong, and Sung Ju Hwang. Chain of retrieval: Multi-aspect iterative search expansion and post-order search aggregation for full paper retrieval. *arXiv preprint arXiv:2507.10057*, 2025. [3, 29, 34]

Revanth Gangi Reddy, Pradeep Dasigi, Md Arafat Sultan, Arman Cohan, Avirup Sil, Heng Ji, and Hannaneh Hajishirzi. ReFIT: Relevance feedback from a reranker during inference. *arXiv preprint arXiv:2305.11744*, 2023. [34]

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and Luke Zettlemoyer. ReasonIR: Training retrievers for reasoning tasks. In *Second Conference on Language Modeling*, 2025. URL https://arxiv.org/abs/2504.20595. [5, 6, 19]

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24497–24524, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1258. URL https://aclanthology.org/2025.findings-acl.1258/. [6]

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 1102–1121, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.71. URL https://aclanthology.org/2023.findings-acl.71/. [1, 33]

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.12883. [6]

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ. [6]

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/. [5, 6, 18]

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *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/. [34]

Jingjin Wang and Jiawei Han. PropRAG: Guiding retrieval with beam search over proposition paths. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6212–6227, 2025. [34]

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. *Advances in Neural Information Processing Systems*, 38: 59888–59915, 2025. [1]

Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. ColBERT-PRF: Semantic pseudo-relevance feedback for dense passage and document retrieval. *ACM Transactions on the Web*, 17 (1), 2023. doi: 10.1145/3572405. URL https://doi.org/10.1145/3572405. [33]

Zehua Wang, Zhaojin Zhang, Boyu Qiu, Xiaolong Weng, Ying Xiong, Buzhou Tang, and Min Zhang. SR-RAG: Verifiable multi-hop reasoning via on-the-fly symbolic graph construction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 38588–38606, 2026. doi: 10.18653/v1/2026.findings-acl.1922. URL https://aclanthology.org/2026.findings-acl.1922/. [34]

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. BrowseComp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025. URL https://arxiv.org/abs/2504.12516. [6]

Orion Weller, Benjamin Van Durme, Dawn Lawrie, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. Promptriever: Instruction-trained retrievers can be prompted like language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.11136. [5, 19, 33]

Wenhan Xiong, Xiang Lorraine Li, Srini Iyer, Jingfei Du, Patrick Lewis, William Yang Wang, Yashar Mehdad, Wen-tau Yih, Sebastian Riedel, Douwe Kiela, and Barlas Oğuz. Answering complex open-domain questions with multi-hop dense retrieval. In *The Ninth International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.12756. [1, 3, 28, 30, 33]

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024. URL https://arxiv.org/abs/2401.15884. [34]

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, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/. [5, 18]

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.03629. [34]

HongChien Yu, Chenyan Xiong, and Jamie Callan. Improving query representations for dense retrieval with pseudo relevance feedback. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021. doi: 10.1145/3459637.3482124. URL https://doi.org/10.1145/3459637.3482124. [1, 3, 28, 33]

Zeta Alpha. NanoBEIR. Hugging Face dataset collection, 2024. URL https://huggingface.co/collections/zeta-alpha-ai/nanobeir-66e1a0af21dfd93e620cd9f6. [6]

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. End-to-end beam retrieval for multi-hop question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1718–1731, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.96. URL https://aclanthology.org/2024.naacl-long.96/. [33]

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025. URL https://arxiv.org/abs/2506.05176. [1, 6]

Chen Zhao, Chenyan Xiong, Jordan Boyd-Graber, and Hal Daumé III. Multi-step reasoning over unstructured text with beam dense retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4635–4641, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.368. URL https://aclanthology.org/2021.naacl-main.368/. [1, 3, 28, 30, 33]

Yilun Zhao, Jinbiao Wei, Tingyu Song, Siyue Zhang, Chen Zhao, and Arman Cohan. Rethinking reasoning-intensive retrieval: Evaluating and advancing retrievers in agentic search systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 36776–36806, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1705. URL https://aclanthology.org/2026.acl-long.1705/. [3, 6, 8, 28]

Andrew Zhu, Alyssa Hwang, Liam Dugan, and Chris Callison-Burch. FanOutQA: A multi-hop, multi-document question answering benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 18–37, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-short.2. URL https://aclanthology.org/2024.acl-short.2/. [5, 6]

Guide to the supplementary material. The supplement details evaluation settings, training data, contextual response, search scaling, the global readout, and downstream reranking. Primary analyses use raw cosine scores and level-balanced aggregation.

Guide to the anonymous website. We encourage readers to also visit the anonymous website. It illustrates the methods of this paper with animations, documents the code, compares rankings side by side, and lets readers browse the training data:

- **Data viewer.** 50 samples for each of our splits, shown in full and parsed into instruction, query, appended observations, target, and hard negatives. Our splits include the company worlds (Company-List, Company-Simple), the persona-seeded web worlds (Seeded-Multi-Hop), the high-recall and knowledge-graph pipelines (high-recall QA, KG fact-checking, KG multi-hop), the evidence chains compiled from multi-hop QA and fact-checking, and the context-lifted Chat-ReasonIR, Doc-ReasonIR, and Doc-Promptriever, each with its recovery variant (e.g., Doc-Promptriever with recovery). Besides the samples, the viewer contains the construction of our synthesized data: the prompts and the intermediate outputs of each synthesis stage.
- **Documentation.** The `rdr` package from the quickstart to the API reference, which is generated from the code: the state rendering, the instructions, every search strategy (tree, chain, beam, breadth, document-as-query, joint feedback), and every readout (global, realized paths, rank fusion, interpolation and Assembly, recovery fusion), each with its parameters, its equation next to the code, and an animation on a toy tree.
- **Code.** The package, its tests, and one reproduction configuration per group of tables.
- **Combinations.** The search and readout strategies of Figure 2, animated, with a configurator that returns the package call and the paper's numbers for each measured combination.
- **Retrieval comparison.** For example queries of each suite, the rankings of Qwen3-Embedding-8B's single lookup and RDR's retrieval tree side by side, with the gold evidence marked.

A Limitations

RDR trades additional query-side encoding for contextual retrieval. Instructions specify the retrieval objective, and performance varies across benchmarks. We set the readout temperature and candidate depth for RDR and did not retune them for other encoders. Multi-hop evaluation uses the benchmark-provided corpora (11.7k–100k documents) and TopiOCQA, a shared 1M-passage corpus, all with exact search; full-Wikipedia indices, approximate search, and a comparison with LLM-driven query decomposition remain open. Differences from GRITHopper include the backbone, training data, and objective. Training at this scale comprises one RRW run; multi-seed estimates remain future work.

B Evaluation and inference specification

RDR-8B uses raw cosine scores, sparse softmax at $\tau = 0.05$ over the top 200 retained candidates per state, and equal total mass per nonempty level (Equation 1). Exclusions are applied before per-state softmax. Single-hop corpora are document-level: we score their level mean directly, without a final support cap. Multi-hop views retain their top 200 aggregated chunks, then take the maximum score for each document, without renormalizing. Candidates outside this final sparse view have zero mass in that view. The per-state operator and level weights are shared; this output processing depends on the corpus representation. Optional recovery fusion and Assembly are identified separately; off-the-shelf baselines retain their model-native input formatting.

B.1 Benchmarks, labels and aggregation

Multi-hop evaluation uses the specified relevance labels and aggregates results by query count: MuSiQue has 1,000 queries (*support*), BrowseComp+ 830 (*evidence*), FanOutQA 310, and FRAMES 813 (*source_group*). MuSiQue's queries are a fixed random sample of its 2,417 answerable validation questions, with the union of their paragraphs as corpus; BrowseComp+ uses the official queries and corpus; FanOutQA uses the dev questions over 700-token chunks of their Wikipedia source pages; FRAMES uses 813 of its 824 questions over chunked source pages. BrowseComp+'s alternate *gold* labels are not summed with evidence. CoIR's main tree score uses the complete official ten-dataset aggregate. BRIGHT covers twelve tasks, NanoBEIR thirteen, and BRIGHT-Pro seven. TopiOCQA uses 2,514 evaluated queries over the shared 1M-passage corpus; scores are aggregated in proportion to query counts. Each injected document is capped at 512 tokens, per the benchmark's protocol on BrowseComp+ (Chen et al., 2026b).

ToolRet averages its metric equally across the 35 task groups. CoIR uses the official ten-dataset aggregate. These definitions apply equally to the main RDR and baseline columns.

Table 4: Gold structure. Group metrics credit any passage from a relevant source group, with recall computed over gold source groups.

Suite	Queries	Corpus	Median passages	Median groups	Passages/group
MuSiQue	1000	11,656	2	2	1.00
BrowseComp+	830	100,195	6	6	1.00
FanOutQA	310	40,209	160	6	28.16
FRAMES	813	30,894	34	3	12.98

B.2 Learning to benefit from retrieval context

Table 5 compares each model with its own root using the recovery instruction at every context-bearing tree level. Both models use the same benchmark scope, tree geometry, and level-balanced readout while reusing their respective document indices. RDR improves on all five suites; the initialization improves on two and declines on three.

Table 5: Matched root-to-tree comparison, nDCG@10. BRIGHT includes twelve domains; CoIR uses ten-dataset macro averaging over twenty subsplits; ToolRet uses 35-task macro averaging. Means weight the five suites equally.

	Stock Qwen			RDR-8B		
Suite	Single-step	Tree	Δ	Single-step	Tree	Δ
BRIGHT	23.17	24.99	+1.82	29.02	30.84	+1.82
NanoBEIR	71.74	70.86	-0.88	66.55	68.13	+1.58
ToolRet	44.72	44.97	+0.25	46.45	48.04	+1.59
CoIR	79.93	78.48	-1.45	78.77	78.88	+0.11
TopiOCQA	59.40	52.89	-6.51	63.37	70.79	+7.42
Mean	55.79	54.44	-1.35	56.83	59.34	+2.50

On TopiOCQA, all evaluated depths are prefixes of a single traversal per model (Table 6). RDR improves after the first contextual level and then remains near that level, while the initialization declines progressively. At $H = 4$, the difference in root-to-tree gains is 13.93 points; the absolute difference between the two tree scores is 17.91.

Table 6: TopiOCQA prefix ladder on 2,514 queries over a 1M-passage corpus, query-weighted nDCG@10. H includes the root; every context-bearing level uses recovery. All prefixes come from the same traversal per model.

Prefix	Stock Qwen	RDR-8B
Single-step	59.40	63.37
$H = 2$	56.72	70.75
$H = 3$	54.13	70.81
$H = 4$	52.89	70.79

B.3 BrowseComp+ relevance labels

BrowseComp+ ships two qrel sets, and we use `qrel_evidence` throughout, matching its official evaluator. It has 5064 positives over 830 queries, a mean of 6.10 per query, so the per-query Recall@k oracle is $\min(k, |G(q)|)/|G(q)|$. A benchmark-level oracle must average that quantity over queries, not divide k by the mean gold count. `qrel_golds` has 2407 positives, 2.90 per query, and is a strict subset of evidence. All BrowseComp+ results use evidence qrels.

B.4 Rendering and context budgets

The deep single-hop tree uses a 4096-token joint query budget; the shallow single-observation control uses 2048. Multi-hop uses a larger joint budget and the stated 512-token cap per injected document. Thus, equal encoder calls do not imply equal processed tokens.

B.5 Instructions

Root instructions follow each benchmark's convention; the multi-hop suites share one root instruction. Context-bearing levels use RDR's update instruction for the suite's role: recovery for relevance refinement, continuation for evidence completion (Section 3). Both update instructions and the multi-hop root instruction are the strings of the training mixture and are shared across suites. The optional recovery view of multi-hop trees (Appendix E.1) uses the wordings listed separately. Table 7 lists all strings verbatim, including the benchmarks' own spelling.

Table 7: Instructions used at evaluation, verbatim. Updated instructions apply at every context-bearing level; root instructions are the benchmarks' own. {domain} is the name of one of the seven StackExchange domains (Biology, Earth Science, Economics, Psychology, Robotics, Stack Overflow, Sustainable Living).

Suite	Instruction
Update instructions (levels $h \geq 2$)	
Recovery Single-Hop: BRIGHT, NanoBEIR, ToolRet, CoIR, TopiOCQA	The previously retrieved document is a distractor or partial observation. Given the query and this document, retrieve the document that fully satisfies the information need.
Continuation: MuSiQue, BrowseComp+, FanOutQA, FRAMES, BRIGHT-Pro	Given the question and the documents already retrieved, retrieve the next document needed to continue answering it.
Recovery Multi-Hop, MuSiQue, BrowseComp+, FanOutQA and FRAMES (Table 20)	The previously retrieved document is a distractor. Given the question and this incorrect document, retrieve the correct document that continues the chain.
Root instructions (level 1)	
BRIGHT, StackExchange domains	Given a {domain} post, retrieve relevant passages that help answer the post
BRIGHT Pony	Given a Pony coding problem, retrieve relevant passages that help answer the problem
BRIGHT LeetCode	Given a coding problem, retrieve relevant examples that help answer the problem
BRIGHT AoPS, TheoremQA questions	Given a Math problem, retrieve relevant examples that help answer the problem
BRIGHT TheoremQA theorems	Given a Math problem, retrieve relevant theorems that help answer the problem
NanoArguAna	Given a claim, find documents that refute the claim
NanoClimateFEVER	Given a claim about climate change, retrieve documents that support or refute the claim
NanoDBPedia	Given a query, retrieve relevant entity descriptions from DBPedia
NanoFEVER	Given a claim, retrieve documents that support or refute the claim
NanoFiQA2018	Given a financial question, retrieve user replies that best answer the question
NanoHotpotQA	Given a multi-hop question, retrieve documents that can help answer the question
NanoMSMARCO	Given a web search query, retrieve relevant passages that answer the query
NanoNFCorpus	Given a question, retrieve relevant documents that best answer the question
NanoNQ	Given a question, retrieve Wikipedia passages that answer the question
NanoQuoraRetrieval	Given a question, retrieve questions that are semantically equivalent to the given question
NanoSCIDOCS	Given a scientific paper title, retrieve paper abstracts that are cited by the given paper
NanoSciFact	Given a scientific claim, retrieve documents that support or refute the claim
NanoTouche2020	Given a question, retrieve detailed and persuasive arguments that answer the question
CoIR codetrans-dl, codetrans-contest	Given a piece for code, retrieval semantically similar code
CoIR codeSearchNet-CCR	Given a code comment, retrieve the code snippet corresponding to that comment.
CoIR codeSearchNet (6 languages)	Given a code snippet, retrieve the comment corresponding to that code.
CoIR apps	Given a question about code problem, retrieval code that can solve user's problem
CoIR cosqa, stackoverflow-qa, codefeedback-st, codefeedback-mt	Given a question about coding, retrieval code or passage that can solve user's question
CoIR synthetic-text2sql	Given a user's question, retrieve SQL queries that are appropriate responses to the question
ToolRet	The instruction of each query, from the dataset (fallback: Retrieve the most relevant tool.)
TopiOCQA	Given a conversation, retrieve the next turn
BRIGHT-Pro	Instruct: Given a {task} post, retrieve relevant passages that help answer the post
MuSiQue, BrowseComp+, FanOutQA, FRAMES	Retrieve the first document needed to begin answering this multi-hop question.

C Training data and contextual-response controls

C.1 Construction families and provenance

We group RRW samples by their construction procedure. Each family can contain standard query–target examples, context-conditioned examples, and recovery variants. Recovery samples retain their source family, so recovery counts overlap with the family totals.

Table 8 defines six construction groups. Existing retrieval datasets contribute both original query–target samples and contextual derivatives. Existing evidence-chain sources include MuSiQue, HotpotQA, and 2WikiMultiHopQA (Trivedi et al., 2022; Yang et al., 2018; Ho et al., 2020), and HoVer and EX-FEVER (Jiang et al., 2020; Ma et al., 2024). Contextual transformations build on

Table 8: RRW construction groups, their principal evidence-retrieval objectives, and their numbers of training samples. Recovery samples keep their source group; Figure 3 shows the same groups.

Family	Construction	Emphasis	Samples	Recovery
Existing retrieval supervision	Query–target pairs and their corrective derivatives	Relevance	2,426,891	554,722
Existing evidence chains	Multi-hop QA and fact verification, compiled into prefix–target transitions	Chains	1,522,431	351,386
Contextual reasoning and instructed retrieval	Document-, dialogue-, or constraint-conditioned versions of existing retrieval tasks	Contextual relevance	293,856	65,852
Company worlds	Company-List and Company-Simple: seeded facts, backstories and linked artifacts	Coverage	21,504	4,899
Web multi-hop	Seeded-Multi-Hop and Constraint-Web: seeds, questions, blueprints and document realization	Chains / constraints	319,356	29,168
Other constructed evidence	High-recall, deep-research and KG/fact-checking pipelines	Mixed	165,686	37,447
Total			4,749,724	1,043,474

reasoning supervision (Shao et al., 2025) and instructed retrieval (Weller et al., 2025). Table 9 lists their named variants; ordinary Promptriever pairs belong to existing retrieval supervision, whereas Doc-Promptriever is a contextual transformation.

Table 9: Contextual transformations in RRW and their training sample counts. Input examples of our splits are shown in the data viewer of the anonymous website; source documentation will accompany the release.

Family	State → target	Ordinary	Recovery
Chat-ReasonIR	dialogue + need → document	40,399	11,945
Doc-ReasonIR	document + question → document	52,939	15,814
Doc-Promptriever	document + constraints → document	134,666	38,093
Promptriever	instruction → document	100,999	30,126

Composition and distinct generators. RRW contains 4,749,724 training samples. Single-hop and multi-hop counts partition this total (Table 10); the two recovery counts sum to 1,043,474 and overlap the partition rather than enlarging it. Company-List and Company-Simple remain distinct generators within Company worlds, as do Seeded-Multi-Hop and Constraint-Web within Web multi-hop. Each generator has its own construction procedure; the final group collects other constructed-evidence pipelines.

Table 10: Structural composition of RRW. Recovery overlaps the single-hop and multi-hop partitions.

Partition or property	Samples	Share
Single-hop samples	2,720,747	57.28%
Multi-hop samples	2,028,977	42.72%
of which recovery	422,900	8.90%
Single-hop recovery	620,574	13.07%
All recovery samples	1,043,474	21.97%

Depth of constructed chains. Constructed chains and evidence sets extend beyond the at most four documents of existing evidence chains. Of the 499,985 constructed samples with a step index, 392,092 target the first to fourth document of a chain or evidence set, 103,192 the fifth to eighth, and 4,701 the ninth to eighteenth; the remaining 6,561 recovery samples carry no step index.

Representative inputs and construction records. Chat-ReasonIR supplies a dialogue and an information need; Doc-ReasonIR supplies a source document and a question; Doc-Promptriever supplies a document and instruction-specific relevance constraints. The target is a useful document.

Recovery counterparts add a corrective `Retrieved Context` observation. The data viewer shows 50 such inputs for each of our splits (e.g., Chat-ReasonIR, Doc-ReasonIR, Doc-Promptriever, and its recovery variant) and the construction records of our synthesized data, e.g., seed, persona, question, source blueprint, layout plan, positive and negative documents, and quality audit for the persona-seeded web worlds.

Corrective views and observed context. A corrective observation can be combined with useful evidence-chain prefixes or context in the original query. One supervised recovery view therefore need not correspond to a single document in the complete input.

C.2 Initialization control and training configuration

Initialization control. We compare RDR-8B with its Qwen3-Embedding-8B initialization under the same search and report retrieval quality and responses to contextual observations relative to each model's root-query performance.

Under the same RDR(4, 3) search, RRW training improves `Recall@5` over the initialization by 17.46, 8.37, 10.13, and 9.16 points on MuSiQue, BrowseComp+, FanOutQA, and FRAMES (Table 1), whereas the tree moves the initialization by at most 2.21 points from its own root.

Contextual retrieval at matched inference cost. We compare each model's gain over its root query when it spends the same inference budget on contextual observations. Table 11 uses the RDR$(2, m)$ operator at the two matched budgets of Table 16. Under the two shallow observation budgets,

Table 11: Gain over each model's root query, `nDCG@10`, from spending the budget on independent observations. m=12 is 13 encoder calls and m=39 is 40, the two budgets of Table 16. Root-query scores are shown for each model; improvements are measured against that model's root. ToolRet uses macro averaging over its 35 tool groups.

	Single-step		Δ at 13 calls		Δ at 40 calls	
Suite	RDR	init	RDR	init	RDR	init
BRIGHT	29.01	23.20	+1.32	-0.14	+1.31	-0.59
NanoBEIR	66.55	71.74	+1.39	-1.05	+1.41	-0.54
ToolRet	46.45	44.72	+1.53	-1.26	+1.33	-1.01
TopiOCQA	63.47	59.43	+3.42	-7.31	+3.63	-6.76

RDR-8B improves over its own root in all eight suite/budget cells and in none with initialization; the largest initialization loss is 7.31 `nDCG@10` on TopiOCQA. The model thus learns to respond to contextual observations from RRW, not from the added inference budget.

Table 12 lists RDR-8B's optimization settings.

Table 12: Training configuration of RDR-8B.

Initialization	Qwen3-Embedding-8B, full fine-tuning
Loss	InfoNCE, temperature 0.02, cosine similarity, last-token pooling
Negatives	one hard negative per sample, in-batch negatives across devices
Optimizer	learning rate 10^{-5}, weight decay 0.01
Schedule	warmup–stable–decay: 10% / 70% / 20% of steps, decay to 0
Batch	2,048 (gradient-cache chunk 4 × 16 accumulation × 32 A100)
Steps	2,319 (one epoch over 4,749,724 samples)
Lengths	queries 4,096, passages 2,048 tokens
Precision	bf16, FlashAttention-2, FSDP full sharding, seed 42

C.3 GRITHopper with RDR's tree and readout

Table 1 evaluates GRITHopper-7B with its own recursive procedure. The tree and the global readout are part of RDR's contribution; for completeness, Table 13 also applies them to GRITHopper-7B's encoder and document index. They improve GRITHopper on the two chained-evidence suites (+1.51 and +1.29 `Recall@5` on MuSiQue and BrowseComp+) but not on the coverage suites (−3.33 and −2.74 on FanOutQA and FRAMES). RDR stays ahead on all four suites: by 0.08 [−1.12, 1.31] on

MuSiQue, GRITHopper's training domain, and by 6.1– 11.0 points on the three suites with larger evidence sets (all 95% intervals above zero).

Table 13: GRITHopper-7B with its own recursive procedure and with RDR's tree and readout, Recall@5/@10.

	MuSiQue		BrowseComp+		FanOutQA		FRAMES	
Model / configuration	R@5	R@10	R@5	R@10	R@5	R@10	R@5	R@10
GRITHopper-7B, recursive	78.55	86.87	12.98	18.04	31.96	43.83	74.08	83.51
GRITHopper-7B, RDR tree and readout	80.06	**88.31**	14.27	20.24	28.63	40.23	71.34	81.58
RDR(4, 3)	**80.14**	87.41	**25.28**	**33.38**	**37.78**	**57.07**	**77.41**	**86.28**

D Depth, breadth and query-side efficiency

The shallow-breadth series retains its original single-observation configuration. Its paired root scores are 28.97, 66.55, 46.42, 78.54, and 63.49 on BRIGHT, NanoBEIR, ToolRet, CoIR, and TopiOCQA, respectively (mean 56.79). Gains in Figure 4 use this root, not the separate recovery-tree root in Table 2.

Two-level special case. With root distribution p_{q_0} and independently searched contextual children p_{v_j}, Equation 1 becomes

$$r(d) = \tfrac{1}{2} p_{q_0}(d) + \tfrac{1}{2} \cdot \frac{1}{m} \sum_{j=1}^{m} p_{v_j}(d). \tag{3}$$

There are $1 + m$ encodings and full-index lookups. Parallelism reduces the number of dependent rounds, not total work.

Table 14 reads cumulative level-balanced aggregation over prefixes of the same $b = 3$ tree; the tree is forward-only, so the H=2, 3, 4 rows are one run read at increasing depth. H counts encoded levels including the root.

Table 14: Single-hop geometry control with the continuation (next-document) instruction: aggregation over prefixes of the same $b = 3$ tree, nDCG@10, separate from the recovery tree of Table 2. The matched recovery-only chat ladder is in Table 6.

H (max. states)	BRIGHT	NanoBEIR	ToolRet	TopiOCQA
2 (4)	30.90	67.65	47.73	68.32
3 (13)	31.09	**68.05**	48.07	69.89
4 (40)	**31.12**	67.96	**48.15**	**70.20**

D.1 Matched maximum state budgets

The multi-hop ladder compares deep and wide configurations with ceilings of 13 and 40 states under the same global readout. Within each pair, depth improves Recall@5 on all four suites. Eligibility and deduplication can reduce realized frontiers; equal ceilings need not imply equal executed calls, latency, or token exposure.

Table 15: Multi-hop geometry ladder, global readout, Recall@5 / Recall@10, All queries, raw cosine scores; maximum states in parentheses. Deep and wide configurations are paired by their maximum state budgets.

Suite	RDR(2, 3) (4)	RDR(3, 3) (13)	RDR(2, 12) (13)	RDR(4, 3) (40)	RDR(2, 39) (40)
MuSiQue	75.90 / 82.23	79.91 / 87.24	71.23 / 78.96	80.14 / 87.41	66.92 / 75.25
BrowseComp+	21.78 / 29.76	24.07 / 32.27	20.38 / 28.05	25.28 / 33.38	19.41 / 26.73
FanOutQA	33.19 / 50.10	36.34 / 54.49	32.77 / 49.20	37.78 / 57.07	31.29 / 47.46
FRAMES	76.05 / 84.49	78.15 / 86.44	73.30 / 83.39	77.41 / 86.28	70.12 / 81.44

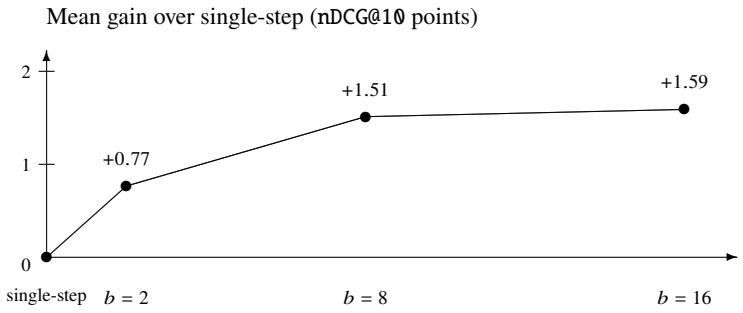


Figure 4: **Shallow breadth scaling without retraining.** Equal-weight mean gain across BRIGHT, NanoBEIR, ToolRet, CoIR, and TopiOCQA for the shallow series in Table 2. Children are searched separately with the shared raw-score aggregation at $H = 2$; gains use the series' paired root specified in Appendix D. BRIGHT-Pro uses a different metric and is reported separately. Markers are measured configurations; connecting lines guide the eye.

The same control on single-hop retrieval. Single-hop breadth uses the shallow RDR(2, m) operator of Table 2: the root and m independent observations, searched separately and combined by the same level-balanced rule, so m observations cost $1 + m$ encoder calls. Matching the two tree budgets gives $m = 12$ against RDR(3, 3) at 13 states and $m = 39$ against RDR(4, 3) at 40. Table 16 reports the three suites whose root scores are identical in both controls; ToolRet and CoIR use a different task aggregation in the breadth runs and are omitted rather than rescaled. The ordering is the same

Table 16: Equal maximum state budgets on single-hop retrieval, nDCG@10. Depth columns are prefixes of the continuation-instruction $b = 3$ tree; breadth columns are the RDR(2, m) operator at the matched maximum budget. Depth uses a 4096-token query budget because a level-four state carries three retrieved documents; breadth carries one and uses 2048. These are separate geometry controls, not replacements for the recovery-only main-table results.

		Up to 13 states		Up to 40 states	
Suite	single-step	wide m=12	deep RDR(3, 3)	wide m=39	deep RDR(4, 3)
BRIGHT	28.97	30.33	**31.09**	30.32	**31.12**
NanoBEIR	66.55	67.94	**68.05**	67.96	67.96
TopiOCQA	63.49	66.89	**69.89**	67.10	**70.20**

as on multi-hop, but the margin is not. Depth leads by 0.00 to 0.80 on BRIGHT and NanoBEIR, against 5.9 to 13.2 Recall@5 on the multi-hop suites, and breadth saturates early: moving from eight to thirty-nine observations changes BRIGHT by +0.16 and NanoBEIR by 0.00. TopiOCQA is the exception, increasing from 3.0 to 3.1 nDCG@10, showing a stronger benefit from accumulated retrieval context on this conversational task without identifying dialogue turns with evidence hops. Two regimes therefore behave differently under the same operator: where evidence must be chained, accumulated context is worth its sequential cost; where one refinement suffices, a single wide step recovers nearly all of the gain at a fraction of the latency, because its m observations are encoded in one batch and searched in one batched operation rather than in H dependent rounds.

D.2 Operation counts and wall-clock time

Table 17 counts query-side operations. N denotes the actual eligible non-root recovery contexts, not an assumed constant. Assembly adds ten dependent encodings against the gathered pool; these are not ten additional full-index searches. A two-level shallow search has two dependent rounds, while RDR(4, 3) has four.

Wall-clock cost of the same prefixes. Because the tree is forward-only, the prefixes of one RDR(4, 3) run are the shallower configurations, so their cost is the cumulative per-level cost of that run. Table 18 reports it as measured on MuSiQue (960 queries after excluding five warm-up queries per shard,

Table 17: Query-side work, excluding the cached document index. N is the number of eligible recovery locations. Primary counts assume a complete $b = 3, H = 4$ tree.

Mode	Encodings	Full-index lookups	Additional selection
Single-step	1	1	None
Single-hop m views	$1 + m$	$1 + m$	None
Primary tree	40	40	Global score reduction
Tree + recovery ($m = 1$)	$40 + N$	$40 + N$	Optional fusion
Tree + Assembly	40 + 10	40	Ten pool-restricted steps

one process per A100-80GB with no co-located job, the states of a level encoded as one batch, CUDA-synchronized timing with model and index loading excluded, every state a full forward pass without KV-caching, exact GPU search over the cached index). Search is negligible throughout;

Table 18: Query-side cost per question on MuSiQue and the Recall@5 it buys (raw cosine scores). State counts are theoretical maxima; times are measured. GRITHopper is bidirectional and is timed and scored through the same tree configuration; its native-protocol score is in Table 1.

Configuration	Max. states	Encode (ms)	Search (ms)	R@5
RDR(1, 0) (single-step)	1	30	0.10	58.51
RDR(2, 3)	4	99	0.25	75.90
RDR(3, 3)	13	364	0.41	79.91
RDR(4, 3)	40	1020	0.59	80.14
Chain (H=4, b=1), global readout	4	151	0.39	80.72
GRITHopper-7B, same tree	40	1079	0.61	80.06

the cost is encoding, and it grows with depth because each level appends a full document to the query (per-level encode means 30, 68, 265, and 656 ms). This holds on larger indices as well: on TopiOCQA's 1M-passage index (fp16, 8.2 GB, exact GPU search), RDR(4, 3) spends 30 ms per query on search and 2.28 s on encoding, and on BrowseComp+ (100k documents) 2.9 ms and 2.95 s, because deeper states carry up to three retrieved documents. RDR(3, 3) delivers 79.91 for 0.36 s, within 0.23 of the 1.02 s RDR(4, 3), and the first level of context (RDR(2, 3), 0.10 s) already accounts for most of the gain over the root. Under the same tree, GRITHopper-7B requires 1.08 s. A single chain with the global readout encodes four states in 0.15 s and reaches 80.72 Recall@5.

E Readout and recovery

We compare five outputs of the same RDR(4,3) traversal. Let E be the documents selected during traversal. A ranks E by path-product scores; C is the primary global readout. B restricts C to $E \cap \mathrm{supp}(C)$ without changing the scores; selected documents outside C's final sparse support are omitted, not zero-filled. A+R and C+R add the same recovery view through z-agreement (Appendix E.1.1). A+R ranks $E \cup \mathrm{supp}(R)$, so it can add candidates rather than only rerank E. Table 19 reports all five outputs.

Table 19: Readout and recovery on a shared raw-score traversal, Recall@5. $I = \Delta_A - \Delta_C$ compares recovery gains from the two baselines. Intervals use 2,000 paired query-bootstrap samples; means weight suites equally.

Readout	MuSiQue	BC+	FanOutQA	FRAMES	Mean
A: realized path ranking	75.97	21.63	36.44	76.71	52.69
B: C restricted to selected documents	80.14	25.28	38.00	77.43	55.21
C: global readout	80.14	25.28	37.78	77.41	55.15
A+R	79.91	24.24	41.68	79.18	56.25
C+R	81.45	25.28	40.66	77.89	56.32
Δ_A	+3.94	+2.61	+5.24	+2.47	+3.57
Δ_C	+1.31	+0.00	+2.88	+0.49	+1.17
I	+2.63	+2.61	+2.36	+1.98	+2.40
95% CI for I	[1.44,3.83]	[1.67,3.60]	[0.65,4.09]	[0.83,3.12]	—

Recovery shows a larger `Recall@5` benefit over the realized baseline across all four suites. At `Recall@10`, the mean interaction decreases to 1.48 points; the suite-level values are 2.54, 1.37, 2.04, and −0.01, respectively, with the FRAMES interval [−0.87,0.81] including zero. The same-score support comparison B versus C changes mean `Recall@5` by only −0.06 points but increases `Recall@100` by 2.96, 7.62, 9.55, and 3.39 points (mean 5.88). Thus, the top-rank benefit is largely retained when output is restricted to selected evidence, while additional candidates improve recall at larger cutoffs. A versus B changes both scoring and sparse support; B versus C holds scores fixed and varies output support.

E.1 Recovery views on the primary traversal

Recovery uses the same encoder with a different instruction, not a second learned operator (🌐 animation). On multi-hop trees, the recovery view renders the last observation as "Retrieved (incorrect)" after its ancestors; it uses the chain-recovery wording described in Table 7. On single-hop, it drives the primary tree; on a continuation tree, it can provide additional rankings of the same observed contexts. A complete RDR(4, 3) tree has 39 non-root contexts, but actual recovery eligibility depends on the generating configuration.

E.1.1 Z-agreement fusion

Let $r_v(d)$ be the aggregated score of document d in view $v \in \{\text{primary}, \text{recovery}\}$, with scored support S_v and $n_v = |S_v|$. For finite scores and $n_v \geq 2$, we standardize each view on its own support:

$$\mu_v = \frac{1}{n_v} \sum_{d \in S_v} r_v(d), \tag{4}$$

$$\sigma_v = \sqrt{\frac{1}{n_v - 1} \sum_{d \in S_v} (r_v(d) - \mu_v)^2}, \tag{5}$$

$$z_v(d) = \frac{r_v(d) - \mu_v}{\widetilde{\sigma}_v}, \qquad \widetilde{\sigma}_v = \begin{cases} 1, & \sigma_v = 0, \\ \sigma_v, & \text{otherwise.} \end{cases} \tag{6}$$

On $S = \bigcup_v S_v$, a missing document receives that view's minimum standardized score:

$$\widehat{z}_v(d) = \begin{cases} z_v(d), & d \in S_v, \\ \min_{u \in S_v} z_v(u), & d \notin S_v. \end{cases} \tag{7}$$

The final score is the equal-weight sum

$$s_{\text{agree}}(d) = \widehat{z}_{\text{primary}}(d) + \widehat{z}_{\text{recovery}}(d). \tag{8}$$

Fusion changes the final ranking while retaining the primary traversal.

The recovery views ranked alone. The primary levels and the auxiliary recovery views are scored over one traversal so that the recovery contribution can be read out on its own. The traversal itself is unchanged in every row of Table 20: expansion is always driven by the continuation instruction, and only the final ranking differs. For RDR-8B, the recovery views alone are competitive with the primary levels, exceeding them by 0.33 on MuSiQue and 2.53 on FanOutQA while losing 0.55 and 1.19 on the other two, so the auxiliary view is not a small correction on top of the tree but a comparable ranking of the same states.

Table 20: Readouts over one RDR(4, 3) traversal, `Recall@5`; primary + recovery uses the main z-agreement fusion. "Recovery alone" ranks only the recovery views; the primary levels still decide which states are expanded.

Readout	MuSiQue	BrowseComp+	FanOutQA	FRAMES
realized b=3	75.90	21.59	36.50	76.71
primary levels	80.14	**25.28**	37.78	77.41
recovery alone	80.47	24.73	40.31	76.22
primary + recovery	**81.45**	**25.28**	**40.66**	**77.89**

F Search and readout on a shared traversal

This appendix separates the readout from the search; the anonymous website animates the compared readouts on a toy traversal (realized paths, global readout). Each comparison reads out one saved traversal, so the states and the documents it retrieves and selects are identical across rows; only the scoring changes. The step from A to B isolates scoring the selected documents with all states instead of their own path, and the step from B to C isolates the candidates that the search never selected, so differences between rows cannot stem from the search. On one saved RDR(4, 3) traversal per suite (2,953 queries; the primary readout reproduces Table 1 exactly), we rank the documents selected during traversal (E) by their path product (A) and by the level-balanced readout over all states (B), and compare both with the complete support (C; Table 21), reporting paired bootstrap intervals (2,000 resamples). Scoring E with all states instead of their paths raises mean `Recall@5` by 2.5 points, with intervals above zero on MuSiQue, BrowseComp+, and FanOutQA (Table 22) and the largest gains on three- and four-hop questions (Table 23); adding the unselected candidates (C) leaves `Recall@5` unchanged and raises mean `Recall@100` by 5.9 points (Appendix E). Removing the root from the readout costs 3.9 and 6.8 points on MuSiQue and FRAMES. MDR's sequence scoring is lower than the path product on three suites and higher on FanOutQA, where outputs of whole chains diversify the evidence set; Assembly (Appendix I.1) targets this case. Rank-based fusion of the same states (reciprocal rank fusion with $k = 60$, and Borda) is 12–18 points below score-based readout. Pooled over the four suites, documents retrieved by more states are more often relevant at every root-score level (Figure 5).

Table 21: Readouts of one RDR(4, 3) traversal, `Recall@5`. Rows A–$B_{-\text{root}}$ rank the selected set E; C ranks the complete support. RRF rows fuse the same states by rank, $k = 60$ (means only).

Readout	MuSiQue	BC+	FanOutQA	FRAMES	Mean
A path product, on E	75.97	21.63	36.44	76.71	52.69
MDR sequence sum, on E	57.55	16.32	45.64	72.14	47.91
B level-balanced mean, on E	80.14	25.28	38.00	77.43	55.21
$B_{-\text{root}}$ (levels ≥ 2), on E	76.27	25.17	39.30	70.62	52.84
C global readout (primary)	80.14	25.28	37.78	77.41	55.15
Flat RRF					40.05
Level-balanced RRF					42.60
Post-order RRF (COR)					41.86
Borda					37.20

Table 22: Step differences in `Recall@5` with paired 95% bootstrap intervals (2,000 resamples, seed 20260921).

Step	MuSiQue	BC+	FanOutQA	FRAMES
$A \to B$	+4.17 [3.02, 5.31]	+3.65 [2.74, 4.56]	+1.55 [0.01, 3.09]	+0.71 [−0.56, 1.90]
$B \to C$	0.00 [0.00, 0.00]	0.00 [0.00, 0.00]	−0.22 [−0.43, −0.05]	−0.02 [−0.06, 0.00]
$B \to B_{-\text{root}}$	−3.88 [−4.84, −2.91]	−0.10 [−0.68, 0.48]	+1.31 [−0.02, 2.66]	−6.81 [−8.00, −5.58]
$A \to$ MDR	−18.42 [−20.43, −16.47]	−5.31 [−6.92, −3.62]	+9.20 [7.13, 11.14]	−4.57 [−6.33, −2.67]

Table 23: MuSiQue by number of gold documents, `Recall@5` ($n = 518/316/166$).

Readout	2 gold	3 gold	4 gold
Path product	90.44	67.62	46.69
MDR sequence sum	64.96	54.11	40.96
$B = C$	89.00	79.01	54.67

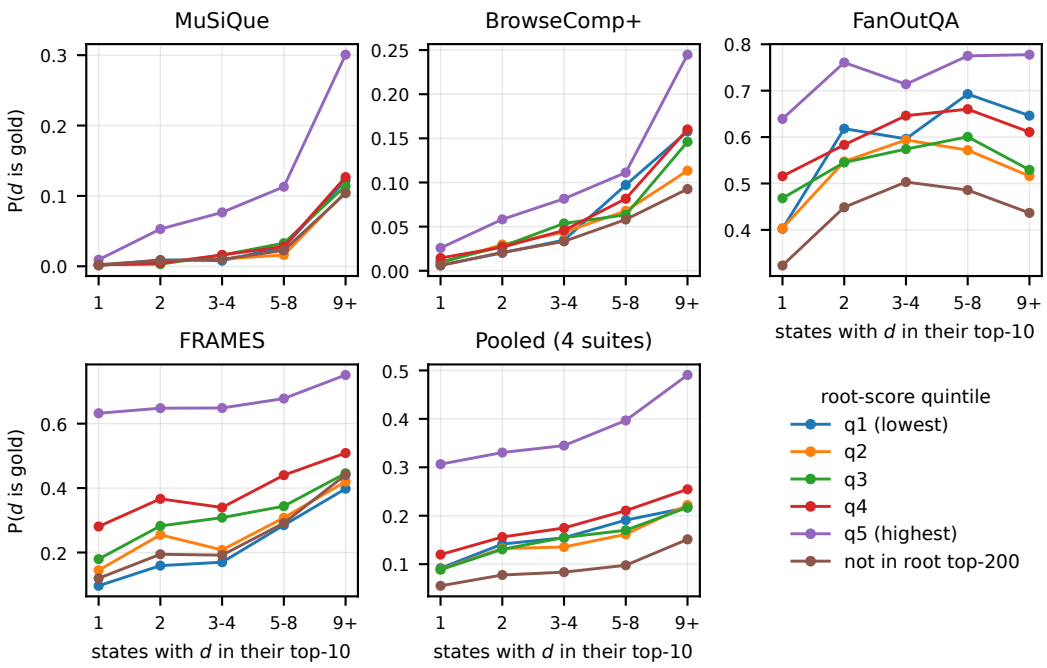


Figure 5: Documents retrieved by more states are more often relevant. P(gold) against the number of states whose top-10 contains the document, per suite and pooled over the four multi-hop suites, one line per root-score quintile plus documents absent from the root's top-200 (pooled: 5.5% → 15.1%).

G Single-hop search and readout

On NanoBEIR and BRIGHT, single-hop readouts follow the multi-hop pattern in both instruction families (Tables 25 and 26). The terminal-state readout, the jointly encoded feedback vector, or MDR's sequence sum falls below the root. The path product improves over the root, and the global readout exceeds the path product by 1.05 and 0.92 points (intervals above zero). Because raw cosine scores are below one, a path product can only shrink along a path. Every document ranks below the documents on its own path, so the root picks lead the list; once the root ranking is complete, it is essentially the root order with a few tree documents inserted. Restricted to the selected documents and backfilled with the root ranking, the level-balanced readout is within 0.2 points of the global readout (intervals include zero). As in multi-hop suites (Appendix F), the top-rank gain therefore comes from scoring the selected documents with all states, not from candidates the search never selected. The single-hop implementation does not deduplicate across branches, so the selected set E contains about six distinct documents per query; readouts restricted to E are therefore reported both on E alone and backfilled with the root ranking.

On TopiOCQA, ToolRet, and a CoIR sample (Tables 27 and 28), the global readout exceeds the path product on every suite, with intervals above zero, most on conversational retrieval (+4.48), where the last hops matter most: because the product keeps parent documents in front, it captures 2.9 of the tree's 7.4 points there. Terminal-state and MDR readouts fall below the root on every suite. The level-balanced readout restricted to E matches the global readout on TopiOCQA and ToolRet. On CoIR, the path product falls below the root (−0.69), the global readout does not (+0.31, interval includes zero), and the unselected candidates add 0.36 points (interval above zero). The CoIR sample draws up to 1,000 queries per subsplit (17,901 queries, official ten-dataset macro); its root and tree scores, 78.56 and 78.87, are close to the full-set 78.77 and 78.88.

The beam and document-expansion configurations of prior work (Table 3a) fall 19–23 nDCG@10 points below the root on the five-suite mean and below it on every suite (Table 24). Beam retrievers output full chains one after another, so the first chain's later hops take ranks that the root's top hits held; a single information need gains nothing from them. Document-as-query expansion replaces the query with document neighbors, which hurts most on CoIR. The wider beam is lower still, because it selects

chains more by their product than by their first hop. The global readout exceeds each configuration by 21.9–25.6 points, with 95% intervals above zero on every suite. These are inference adaptations with RDR's encoder, not the original trained systems. Root interpolation at DC-PRF's weight is the only prior configuration above the root (+0.79 [0.63, 0.96]), mainly on TopiOCQA, and stays 1.77 [1.50, 2.06] below RDR; joint feedback falls 4.08 [3.60, 4.54] below the root. With the global readout, a 13-wide beam reaches 59.60, 0.25 [0.05, 0.45] above the tree, so on single-hop retrieval, the readout, not the search, carries the gain.

Table 24: Prior-work configurations on single-hop retrieval (Table 3a), nDCG@10, recovery instruction, same queries as Table 27. Beam outputs are the realized chains in hop order, completed with the root ranking when they hold fewer than ten distinct documents (at most 4.6% of queries, W=5 only). The root was re-encoded for this run; on CoIR, 10 of 17,901 queries differ in bf16 batching (78.57 versus 78.56). Joint feedback uses the benchmark instruction; DC-PRF's interpolation uses its root weight $\lambda = 0.8$, and $\lambda = 0.5$ and 0.2 are reported without selection.

Configuration	NanoBEIR	BRIGHT	TopiOCQA	ToolRet	CoIR	Mean	Δ root
Root	66.55	28.99	63.37	46.44	78.57	56.78	
as in ANCE-PRF (m=3)	64.60	28.92	46.44	45.62	77.96	52.71	−4.08
as in DC-PRF (λ=0.8)	66.90	29.25	66.25	46.66	78.79	57.57	+0.79
λ=0.5	66.95	29.62	66.93	46.59	78.68	57.75	+0.97
λ=0.2	66.19	29.89	62.50	45.51	77.77	56.37	−0.41
as in BeamDR, W=5	43.41	19.42	29.85	30.59	63.90	37.43	−19.35
as in MDR, W=5	42.42	19.52	26.88	29.97	63.28	36.41	−20.37
as in BeamDR, W=13	39.91	16.49	30.29	28.37	62.55	35.52	−21.26
as in MDR, W=13	38.05	16.31	25.86	27.38	61.34	33.79	−23.00
as in COR	46.96	17.48	47.95	30.28	42.18	36.97	−19.81
Search variants with RDR's readouts (Table 3b)							
chain / global readout	67.21	30.34	66.86	46.91	78.44	57.95	+1.17
beam (W=13) / path product	66.67	28.98	63.38	46.33	78.05	56.68	−0.10
beam (W=13) / global readout	**68.93**	**31.01**	**71.55**	47.94	78.56	**59.60**	+2.81
document-as-query / global readout	64.73	29.27	61.70	41.85	77.60	55.03	−1.75
RDR, global readout	68.13	30.81	70.79	**48.11**	**78.87**	59.34	+2.56

Table 25: Single-hop readouts, task-macro nDCG@10 over NanoBEIR and BRIGHT (25 tasks), recovery and continuation (next-document) instruction. Backfilled: E ranked by the readout, then filled with the root's top-200 in root order, duplicates skipped.

Readout	Recovery	Next-doc
Root	48.52	48.52
Joint feedback (m=3)	–	47.52
Chain, terminal state	47.95	47.40
Path product, on E	46.49	47.33
Path product, backfilled	49.16	49.35
MDR sequence sum, on E	42.55	42.42
MDR sequence sum, backfilled	45.19	44.43
B level-balanced mean, on E	47.46	48.38
B, backfilled	50.11	50.40
Global readout (primary)	50.22	50.27

Table 26: Paired differences in single-hop nDCG@10 (25 tasks) with 95% bootstrap intervals (queries resampled within tasks, 2,000 resamples, seed 20260921). bf.: backfilled with the root ranking.

Comparison	Recovery	Next-doc
Terminal chain − root	−0.57 [−1.38, 0.22]	−1.12 [−1.93, −0.31]
Path product (bf.) − root	+0.64 [0.39, 0.90]	+0.83 [0.48, 1.15]
MDR (bf.) − root	−3.33 [−4.13, −2.48]	−4.08 [−4.94, −3.24]
B (bf.) − root	+1.60 [1.08, 2.11]	+1.89 [1.33, 2.44]
Global − root	+1.70 [1.14, 2.24]	+1.75 [1.17, 2.33]
Global − path product (bf.)	+1.05 [0.56, 1.55]	+0.92 [0.42, 1.41]
Global − B (bf.)	+0.10 [−0.11, 0.31]	−0.14 [−0.32, 0.05]

Table 27: Single-hop readouts per suite, nDCG@10, recovery instruction. Path, MDR, and B readouts are completed with the root ranking. NanoBEIR and BRIGHT: task macro; TopiOCQA: query mean (2,514 queries); ToolRet: 35-group macro (7,961 queries, per-query run); CoIR: ten-dataset macro on the 17,901-query sample. Root and the global readout reproduce Table 2 on NanoBEIR and TopiOCQA; BRIGHT differs by 0.03, and ToolRet's per-query run gives 48.11 versus 48.04.

Readout	NanoBEIR	BRIGHT	TopiOCQA	ToolRet	CoIR	Mean
Root	66.55	28.99	63.37	46.44	78.56	56.78
Chain, terminal state	64.66	29.86	56.47	43.94	76.63	54.31
Path product	67.20	29.63	66.31	47.24	77.87	57.65
MDR sequence sum	60.93	28.14	51.45	44.31	72.56	51.48
B level-balanced mean, on E	68.05	30.68	70.68	48.01	78.50	59.19
Global readout (primary)	68.13	30.81	70.79	48.11	78.87	59.34

Table 28: Paired differences in single-hop nDCG@10 with 95% bootstrap intervals (2,000 resamples, seed 20260921), recovery instruction. NanoBEIR and BRIGHT are pooled as in Table 26; CoIR is the 17,901-query sample. CC: global (complete-child) readout; path: path product.

Comparison	NanoBEIR + BRIGHT	TopiOCQA	ToolRet	CoIR
CC − root	+1.70 [1.14, 2.24]	+7.42 [6.45, 8.32]	+1.67 [1.20, 2.13]	+0.31 [−0.20, 0.85]
Path − root	+0.64 [0.39, 0.90]	+2.94 [2.52, 3.38]	+0.80 [0.48, 1.17]	−0.69 [−0.99, −0.36]
Terminal − root	−0.57 [−1.38, 0.22]	−6.91 [−8.33, −5.59]	−2.50 [−3.30, −1.68]	−1.93 [−2.43, −1.44]
MDR − root	−3.33 [−4.13, −2.48]	−11.92 [−13.12, −10.75]	−2.13 [−2.89, −1.36]	−6.00 [−6.52, −5.47]
CC − path	+1.05 [0.56, 1.55]	+4.48 [3.68, 5.23]	+0.87 [0.40, 1.31]	+0.99 [0.50, 1.49]
CC − B	+0.10 [−0.11, 0.31]	+0.11 [−0.05, 0.26]	+0.10 [−0.10, 0.30]	+0.36 [0.08, 0.66]

G.1 BRIGHT domain groups

The largest original-BRIGHT gains occur on its seven StackExchange domains. Under BRIGHT-Pro's weighted-aspect evaluation (Zhao et al., 2026), RDR reaches 61.13, compared with BGE-Reasoner's 61.82 (Table 2). These are separate benchmark summaries; the grouped original-BRIGHT comparison in Table 29 uses its corresponding root scores.

Table 29: Original BRIGHT, macro nDCG@10. Deltas are computed before rounding. BGE uses an 8192-token limit and RDR a 4096-token limit.

Domain group	RDR root	RDR tree	Δ	BGE root	Tree − BGE
StackExchange (7)	31.94	34.99	+3.05	41.44	-6.45
Coding/theorem (5)	24.94	25.02	+0.09	30.72	-5.70
All BRIGHT (12)	29.02	30.84	+1.82	36.97	-6.13

H Same-encoder inference controls

These controls use RDR-8B and the same cached document index; the anonymous website documents and animates each inference form (joint feedback, chain, beam, document-as-query). Joint feedback follows ANCE-PRF's input form (Yu et al., 2021): the query and first-pass passages are encoded together into one query vector. The greedy chain selects one successor per state; terminal-state and all-level readouts differ in whether earlier rankings contribute to the output. Root interpolation follows DC-PRF's readout (Albishre, 2026): the last chain state is interpolated with the root query as the ℓ_2-normalized $\lambda z_{\text{root}} + (1-\lambda) z_{\text{last}}$, at DC-PRF's root weight $\lambda = 0.8$. Beam-style search follows the path-search lineage of MDR and BeamDR (Xiong et al., 2021; Zhao et al., 2021). These are inference adaptations at a shared encoder, rather than comparisons of their original training procedures. Joint feedback uses the benchmark instruction; the contextual arms use their stated update instructions.

All multi-hop controls in this comparison use raw cosine scores, $\tau = 0.05$, top-200 per-state support, and, where indicated, the level-balanced readout. Table 30 lists the multi-hop inference forms; its primary tree row reproduces Table 1. All differences are measured against the root evaluated in this experiment.

Table 30: Inference forms with the same RDR encoder, Recall@5 (raw cosine scores). Mean weights the four suites equally; tree and Beam13 share a maximum budget of 40 primary states, not a verified equal number of executed calls.

Inference form	MuSiQue	BrowseComp+	FanOutQA	FRAMES	Mean
Root	58.51	16.87	28.63	66.77	42.70
Joint feedback, $m = 3$	49.35	21.14	32.17	49.93	38.15
Chain, terminal state	36.27	17.46	28.98	46.06	32.19
Chain, root-interpolated (λ=0.8, as in DC-PRF)	61.73	18.93	29.42	68.62	44.68
λ=0.5	64.91	20.77	31.29	64.38	45.34
λ=0.2	48.86	19.58	29.91	53.09	37.86
Chain, global readout	80.72	23.13	37.70	74.09	53.91
Beam 5, global readout	80.27	25.60	37.57	76.01	54.86
Beam 10, global readout	79.03	25.00	37.00	76.09	54.28
Beam 13, global readout	78.59	24.32	37.11	75.50	53.88
Tree, global readout	80.14	25.28	37.78	77.41	**55.15**

Interpolating the last state with the root at DC-PRF's weight raises the terminal chain by 12.5 points to 44.68 Recall@5, 1.98 above the root (95% intervals above zero on MuSiQue, BrowseComp+, and FRAMES) and 10.5 below the tree; the other weights were run without selection ($\lambda = 0.5$: 45.34; $\lambda = 0.2$: 37.86). Retaining all levels raises the chain's mean by 21.72 points; branching adds 1.24 over that aggregated chain. The tree improves over the chain on BrowseComp+, FanOutQA, and FRAMES, while the chain leads by 0.58 points on MuSiQue. The tree has the highest primary-readout mean among the tested beam widths, but Beam5 is slightly higher on MuSiQue and BrowseComp+. For the recovery-only view, Beam5 reaches 56.42 versus the tree's 55.44; the primary-readout comparison does not imply dominance across all views.

Table 31: Single-hop inference-form control, task-macro nDCG@10 on the fixed 43-task intersection: BRIGHT (12), CoIR (18), NanoBEIR (13), using the continuation (next-document) instruction; Table 2 uses recovery. Appendix G reports both instruction families on NanoBEIR and BRIGHT. Both planned feedback budgets are shown. This subset is separate from CoIR's official ten-dataset main-table aggregation.

Inference form	BRIGHT	CoIR subset	NanoBEIR	Task mean
Root	29.00	84.35	66.55	63.52
Joint feedback, $m = 1$	29.43	81.31	64.14	61.64
Joint feedback, $m = 3$	29.02	83.20	64.60	62.46
Chain, terminal state	29.38	81.97	63.87	61.82
Tree, global readout	31.12	84.35	67.96	**64.54**

On the single-hop task intersection (Table 31), the tree improves the task mean by 1.02 points, with gains on BRIGHT and NanoBEIR and a neutral result on this CoIR subset. The terminal chain and joint-feedback variants are below the root in aggregate, with effects varying by suite.

Paired uncertainty. Query-paired bootstrap uses 2000 resamples, seed 20260921, within each suite. For tree minus aggregated chain, the Recall@5 differences and 95% intervals are −0.58 [−1.59, 0.48] on MuSiQue, 2.14 [1.09, 3.24] on BrowseComp+, 0.09 [−1.30, 1.51] on FanOutQA, and 3.32 [2.08, 4.50] on FRAMES. MuSiQue's Recall@10 interval is negative [−2.09, −0.47]. Tree minus Beam13 has positive Recall@5 intervals on MuSiQue, BrowseComp+, and FRAMES, but not FanOutQA. Deltas use unrounded query-level scores; significance statements are restricted to the comparisons with reported intervals.

Document-as-query expansion. To separate learned state updates from tree expansion as such, we keep RDR's geometry, eligibility, and deduplication but let each child search with the selected document's own index embedding, as when retrieved papers are promoted to queries (Park et al., 2025); this is a COR-style expansion and aggregation without COR's LLM-generated aspect queries, not a full reproduction. Learned, query-anchored states outperform document-neighborhood expansion on all four suites (Table 32).

Table 32: Learned states versus document-as-query expansion on the same geometry, Recall@5. Differences to RDR with paired 95% bootstrap intervals.

Search / readout	MuSiQue	BC+	FanOutQA	FRAMES	Mean
RDR(4, 3), global readout	80.14	25.28	37.78	77.41	55.15
Doc-as-query, level-balanced mean	57.14	18.01	26.19	60.30	40.41
Doc-as-query, post-order RRF (COR)	40.93	10.27	25.02	44.07	30.08
Δ level mean vs. RDR	−23.00 [−24.50, −21.43]	−7.27 [−8.47, −5.99]	−11.60 [−13.46, −9.95]	−17.11 [−18.62, −15.44]	

H.1 Beam-style search at matched maximum state budget

This control uses the RDR encoder with search inspired by MDR and BeamDR (Xiong et al., 2021; Zhao et al., 2021), rather than their original trained models. We test beam widths $W = 5$ and 13; $W = 13$ permits $1 + 3W = 40$ primary states, matching the ceiling of RDR(4, 3). Table 33 compares raw-cosine readouts of the same beam traversal; recovery costs are additional. The global readout

Table 33: Beam search with RDR-8B (raw cosine), $W = 5$ and 13. Realized outputs of the same beam traversal—the path product, the probability chain as in BeamDR, and the sequence sum as in MDR—versus the global readout, Recall@5. The path product ranks all hop-1 picks first and therefore matches the root in its top positions.

Readout	MuSiQue	BrowseComp+	FanOutQA	FRAMES
Root	58.51	16.87	28.63	66.77
$W = 5$				
Path product	58.89	16.87	28.63	66.83
Probability chain (as in BeamDR)	55.51	15.35	45.09	69.24
Sequence sum (as in MDR)	53.94	13.17	44.41	67.37
Global readout	**80.27**	**25.60**	37.57	**76.01**
$W = 13$				
Path product	58.74	16.87	28.63	66.83
Probability chain (as in BeamDR)	48.17	12.23	41.30	60.81
Sequence sum (as in MDR)	47.12	9.54	40.86	58.41
Global readout	**78.59**	**24.32**	37.11	**75.50**

exceeds both realized chain outputs on MuSiQue, BrowseComp+, and FRAMES at both widths by at least 24.8, 10.3, and 6.8 Recall@5 points (all 95% intervals above zero); on FanOutQA, chain-ordered output is 3.8–7.5 points higher, since whole chains cover more distinct evidence groups.

I Coverage selection from a shared candidate pool

I.1 Assembly: set-conditioned selection

Tree states see their ancestor path, not sibling branches, so that different paths can retrieve redundant evidence. Optional *Assembly* supplies a horizontal, set-conditioned update using the same encoder to favor complementary evidence across paths (🌐 animation). The tree already supports coverage retrieval; Assembly adds a set-selection step. From the top-50 tree pool C, it selects ten documents, conditioning on the already chosen set S_j rather than a single path:

$$d_{j+1} = \arg\max_{d \in C \setminus S_j} \cos\big(f_\theta([\iota_{\text{set}}; q; \text{ctx}(S_j)]), e_d\big), \qquad S_{j+1} = S_j \cup \{d_{j+1}\}. \tag{9}$$

Starting from $S_0 = \varnothing$, this adds ten pool-restricted encoder calls, not full-index searches. Matched-pool controls follow below.

I.2 Matched candidate pools

Root-pool versus tree-pool. Table 34 uses the same model, corpus, metric, traversal mask, top-50 pool, and ten selections. A = single-step top-50 by rank; B = A + Assembly; C = tree top-50 by level-balanced rank; D = C + Assembly (Recall@5 / @10). D is the Assembly row of Table 1; C reproduces its RDR(4, 3) row within 0.3 Recall@5. Assembly works with no tree at all (B−A = +11.81 / +4.53 / +15.70 / +10.56 Recall@5 on MuSiQue / BrowseComp+ / FanOutQA / FRAMES)

Table 34: Matched root/tree candidate pools with and without Assembly, Recall@5 / Recall@10.

	A	B	C	D
MuSiQue	58.51 / 65.99	70.32 / 73.15	80.06 / 87.49	79.62 / 82.86
BrowseComp+	16.87 / 23.10	21.40 / 27.09	24.98 / 33.29	23.63 / 30.23
FanOutQA	28.63 / 42.72	44.33 / 56.81	37.92 / 56.99	49.29 / 63.52
FRAMES	66.77 / 76.83	77.33 / 81.64	77.59 / 86.54	79.20 / 83.43

and the two are largely substitutes; the tree still contributes candidates that the root pool lacks (D−B = +9.30 / +2.23 / +4.96 / +1.87). The orderings follow task geometry: on the chained suites, the tree alone wins, and Assembly costs Recall@5 (MuSiQue and BrowseComp+: C > D > B > A); on the coverage suites, Assembly on the tree pool is best (FanOutQA D > B > C > A, FRAMES D > C > B > A). At Recall@10, Assembly on the tree pool, which returns exactly ten documents, gains only on FanOutQA.

J LLM RERANKING

Listwise LLM rerankers cannot recover evidence absent from their shortlist. Following El Assadi et al. (2026), we compare the same 27B judge over the stock and RDR candidate pools to test whether contextual retrieval improves the available evidence. The stock pool contains Qwen3-Embedding-8B's top-50 for the original query. The RDR pool contains the primary RDR(4,3) tree's top-50 results under raw-cosine scoring and the level-balanced global readout, without auxiliary recovery fusion or Assembly. Both pools use the same width-oriented instruction (Appendix J.1), chosen because all four suites reward recovering several evidence documents, and the same document rendering.

Table 35: 27B listwise reranking of each first stage's top-50, Recall@5, with the same width-oriented prompt. The RDR pool is the primary RDR(4, 3) tree of Table 1, without recovery fusion or Assembly. Pool recall and paired intervals are in Table 36.

Top-50 source	MuSiQue	BrowseComp+	FanOutQA	FRAMES
Stock Qwen3, first stage	63.16	14.70	26.79	67.80
+ 27B reranker	78.02	31.72	48.04	80.86
RDR, first stage	80.14	25.28	37.78	77.41
+ 27B reranker	**86.66**	**41.93**	**54.18**	**83.15**

Table 36: 27B listwise reranking of each first stage's top-50, Recall@5. Pool: Recall@50 of the top-50, an upper bound for any reranker. Last rows: difference after reranking with paired 95% bootstrap intervals.

Top-50 source		MuSiQue	BrowseComp+	FanOutQA	FRAMES
Stock Qwen3	pool R@50	86.45	38.81	78.60	92.65
	first stage	63.16	14.70	26.79	67.80
	27B	78.02	31.72	48.04	80.86
RDR (primary)	pool R@50	95.87	53.22	87.91	95.85
	first stage	80.14	25.28	37.78	77.41
	27B	86.66	41.93	54.18	83.15
RDR − Stock, 27B		+8.64 [7.36, 9.90]	+10.21 [8.59, 11.80]	+6.14 [4.14, 8.06]	+2.29 [1.05, 3.52]

Rerankers complement RDR rather than replace it: RDR's top-50 contains more gold evidence than stock Qwen3's (Recall@50 is higher by 3.2–14.4 points), and reranking it yields higher Recall@5 on all four suites, by 2.3–10.2 points (95% intervals above zero; Tables 35 and 36). On MuSiQue, RDR's first stage alone (80.14) already exceeds the reranked single-query pool (78.02). The reranked margin tracks the pool recall gap (Recall@50 gap: 9.4, 14.4, 9.3, and 3.2 points; margin after reranking: 8.6, 10.2, 6.1, and 2.3).

We follow the listwise protocol of El Assadi et al. (2026): one prompt per query, holding all 50 candidates, documents truncated to 800 characters, candidates ordered by first-stage score, and a

score of $k -$ rank read off a JSON id list. Candidates are the first stage's own top-50, with no gold injection, so `Recall@50` of the shortlist upper bounds every reranked score. On a parse failure, we append the omitted IDs in first-stage order, which biases an affected cell *toward* the retriever rather than toward noise. Reasoning is disabled for the Qwen judge (`enable_thinking=false`).

J.1 Prompts

Table 37 compares the instruction of El Assadi et al. (2026) with our width- and depth-oriented instructions on the two root pools. Width is best on BrowseComp+, FanOutQA, and FRAMES and within 0.9 points of the best instruction on MuSiQue, so we use it for all reported reranking results.

Table 37: Judge instruction for 27B listwise reranking of the top-50 of each root query, `Recall@5`; bold: best instruction per row. The comparison used root pools from an earlier run; on MuSiQue and FanOutQA its Qwen3 pool differs from the one in Tables 35 and 36 (first stage 61.25 and 25.76 versus 63.16 and 26.79).

Pool	Suite	First stage	El Assadi et al.	Width	Depth
Qwen3 root	MuSiQue	61.25	**75.00**	74.17	70.83
	BrowseComp+	14.70	31.56	**31.72**	30.88
	FanOutQA	25.76	50.19	**54.04**	47.51
	FRAMES	67.80	74.60	**80.86**	79.73
RDR root	MuSiQue	58.38	72.11	74.54	**75.02**
	BrowseComp+	16.92	35.13	**35.58**	34.83
	FanOutQA	28.43	44.81	**50.36**	46.03
	FRAMES	66.93	75.25	**81.67**	79.65

El Assadi et al. Verbatim `LISTWISE_INSTRUCTIONS` from the released implementation of El Assadi et al. (2026).

El Assadi et al. listwise instruction

```
You are an expert at evaluating document relevance. Given a query and a list
of candidate documents, rank them by relevance to the query. Output ONLY a JSON
list of integer document IDs in order of decreasing relevance. Example output
format: [12, 3, 47, 1, ...]. Include ALL document IDs from the input.
```

width Ours. The objective is joint coverage of distinct sub-questions rather than per-document similarity.

Width instruction (ours)

```
You are an expert at assembling evidence for multi-part questions. The query
needs several DIFFERENT facts, each usually found in a different document. Rank
the candidates so that the highest-ranked documents TOGETHER cover as many
distinct parts of the query as possible. Prefer a document that adds a part
not yet covered over another document about a part already covered, and push
near-duplicates of already-ranked documents down. Output ONLY a JSON list of
integer document IDs in order of decreasing usefulness for covering the query.
Example output format: [12, 3, 47, 1, ...]. Include ALL document IDs from the
input.
```

depth Ours. The objective is advancing a reasoning chain rather than surface similarity.

Depth instruction (ours)

```
You are an expert at multi-hop reasoning over documents. Answering the query
requires following a chain: an early document supplies an entity or fact needed
to locate the next one. Rank the candidates by how much each ADVANCES that
chain toward the final answer, not by how similar it looks to the query wording.
A document that names the bridging entity ranks above a document that merely
shares vocabulary with the query. Output ONLY a JSON list of integer document
```

```
IDs in order of decreasing usefulness for completing the chain.  Example output
format:  [12, 3, 47, 1, ...].  Include ALL document IDs from the input.
```

Output validation. Judge outputs are checked for missing, duplicate, and invalid IDs; missing IDs are backfilled in first-stage order.

K Detailed Related Work

Among decomposition-free recursive retrievers, MDR and Beam Retrieval train sub-1B encoders on in-domain hop supervision for the benchmark they are evaluated on; GRITHopper-7B is the scale-matched instance trained across benchmarks and is therefore the recursive baseline in Table 1.

Task scope of prior configurations. MDR fixes the chain length to two hops on HotpotQA and multi-evidence FEVER (Xiong et al., 2021), and BeamDR evaluates two-passage chains on HotpotQA (Zhao et al., 2021). Beam Retrieval notes that earlier retrievers were customized for two-hop questions and extends beam scoring to MuSiQue's two- to four-hop setting, scoring passages from a given candidate pool (Zhang et al., 2024). GRITHopper retrieves over a full index and is evaluated on two to four hops, in and out of distribution (Erker et al., 2026). Pseudo-relevance feedback, including DC-PRF, targets single-hop retrieval (Yu et al., 2021; Albishre, 2026). Each configuration in Table 3a was thus designed for one task family; RDR evaluates one operator on single-hop retrieval, on chains of two to four hops, and on evidence sets with a median of up to six source groups (Table 4).

Decomposition-free recursive retrieval. MDR (Xiong et al., 2021), BeamDR (Zhao et al., 2021), and GRITHopper (Erker et al., 2026) re-encode a question with retrieved passages and search over evidence paths without generating subquestions. They establish the recursive mechanism and strong depth retrieval. Their training target remains the next item on a successful document chain, and their readout is the beam: they discard documents a state scored but did not select, restricting returned evidence to selected paths. RDR retains the mechanism, broadens the state distribution to breadth and to failed observations, and aggregates retained scores beyond selected successors. Beam Retrieval (Zhang et al., 2024) learns the beam scoring end-to-end but likewise ranks paths, not the scored population. TreeHop (Li et al., 2025b) updates query embeddings with a gated cross-attention over frozen document embeddings, prunes branches that revisit retrieved chunks, and returns the union of the layer-wise top-K chunks; like the beam retrievers, it selects documents without aggregating scores across states. Earlier multi-step retrievers update the query vector from a reader's state with a learned recurrent step (Das et al., 2019), retrieve reasoning paths over Wikipedia's hyperlink graph with a recurrent retriever and beam search (Asai et al., 2020), or condense the facts retrieved at each hop into the next hop's query (Khattab et al., 2021); each is trained for multi-hop question answering and returns the passages of its selected paths or hops.

Coverage-aware and set retrieval. MMR trades relevance against redundancy (Carbonell & Goldstein, 1998); JPR decodes passage sets from a bounded pool (Min et al., 2021), SetR selects a sufficient set with an LLM (Lee et al., 2025), AMER emits several query embeddings (Chen et al., 2025), and CoveR trains a bi-encoder for nugget coverage (Ju et al., 2026). RDR differs in candidate generation—repeated full-index lookups build a recursively conditioned pool before state-conditioned set selection. Our contribution is the recursively conditioned candidate pool, not diversity reranking or multi-vector retrieval. We compare against the released CoveR checkpoint.

Contextualized queries and instructed embeddings. Instruction-tuned retrievers (Su et al., 2023; Asai et al., 2023; Weller et al., 2025) condition the query embedding on a task description; Contextual Document Embeddings (Morris & Rush, 2025) condition on neighboring corpus examples. Dense pseudo-relevance feedback instead updates the query from retrieved documents: ANCE-PRF jointly encodes the query and several first-pass passages into one learned query vector for a second search over an unchanged index (Yu et al., 2021). ColBERT-PRF adds clustered feedback-token vectors to a multi-vector query (Wang et al., 2023); Li et al. (2023) study vector-based feedback and fusion of separately contextualized cross-encoder rerankings. DC-PRF (Albishre, 2026) sequentially refines a frozen encoder's query vector over the ranked feedback documents with a trained gated recurrent update, then ranks with the final state interpolated with the original query. Table 3a runs this inference form with RDR's encoder: a sequence of refined query states, read out by the last state interpolated with the root at DC-PRF's weight $\lambda = 0.8$. The states are re-encoded with RDR's

encoder instead of a learned gate, so this is an inference analog, not a reproduction. RDR combines an instruction, query, and observed artifacts in one encoder, recursively composing its updates across refinement, continuation, and coverage. Its distinction is this shared contextual operator and search framework, rather than feedback or multiple query views alone. Erker et al. (2024) explored composing independently encoded context artifacts instead of re-encoding them for dialog representations.

Score fusion. Fusing several ranked lists over one index is classical: CombSUM adds normalized scores and CombMNZ additionally multiplies by the number of lists that retrieve the document (Fox & Shaw, 1994); reciprocal rank fusion discards scores (Cormack et al., 2009). Our main readout averages states within each level and then averages the nonempty levels. Aggregation *across levels of a retrieval tree* has been proposed for scientific paper retrieval, where retrieved papers become new queries, and sibling ranked lists are merged by reciprocal rank fusion and folded recursively into their parents (Park et al., 2025). There, the aggregated lists are parallel views with no learned update; RDR aggregates scores from sequentially composed, learned states. On RDR's traversal, this post-order rank fusion is 13 points below the score-based readout, and document-as-query expansion is 7–23 points below learned states (Appendix H). Graph-based retrievers score passages globally by personalized PageRank over an entity or proposition graph extracted offline by an LLM (Gutiérrez et al., 2025; Wang & Han, 2025); the aggregated units are graph nodes, not query states produced by a learned update at test time. Reranker and corpus feedback refine queries or candidate pools from retrieved evidence (Reddy et al., 2023; MacAvaney et al., 2022).

Corrective and agentic retrieval. ReAct, IRCoT, Self-RAG, CRAG, and search-RL systems reformulate, critique, or stop with a language model (Yao et al., 2023; Trivedi et al., 2023; Asai et al., 2024; Yan et al., 2024; Jin et al., 2025). RDR performs instruction-conditioned search updates directly in embedding space.

Symbolic reasoning and global evidence fusion. SR-RAG generates subquestions, constructs a query-specific symbolic graph, and verifies reasoning with a symbolic engine (Wang et al., 2026). It complements the retrieval-stage lineage above: explicit decomposition and verification differ from training RDR's contextual dense operator. RDR combines a learned contextual encoder with recursive search and global readout across heterogeneous retrieval objectives.