UNDERSTAND ONCE, RETRIEVE MANY TIMES: SAADI FOR SINGLE-PASS RETRIEVAL OF DISTRIBUTED EVIDENCE IN LONG DOCUMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-document retrieval is challenging when relevant evidence is spread across distant passages or multiple hops, especially under a limited retrieval budget. Single-pass top-k retrievers have predictable compute and latency at query time, whereas agentic systems pursue higher recall by reading a document over several steps. We move progressive document understanding to index time, where a single-pass top-k retriever can use it without query-time large language model (LLM) calls or adaptive trajectories. SAADI (**S**tateful **A**ccumul**a**tion of **D**ocument **I**nformation) builds a query-agnostic document state at index time with one state-update call per chunk. Each call extracts the chunk's entities, conditioned on records recalled from the entity memory and on a hierarchical summary of the prefix. The chunk is linked to the resulting records and to the summary covering it, and the original chunks remain the evidence and retrieval units. At query time, two auxiliary scores are computed from a chunk's linked records and its covering summary and fused with any sparse or dense base retriever's score in a single top-k pass. Index-time LLM calls, including summary folds, are fixed before any query arrives, at most 1.25 per chunk in the default configuration. Each call reads a context that grows only logarithmically with the document, and appending chunks extends the state without rebuilding it. Across four datasets and five retrievers, SAADI improves HR@10 over its base retriever in all twenty pairs, by 0.3 to 12.4 points (1.1 to 4.0 under BM25). It ranks first or second in 34 of 40 dataset, retriever, and depth cells and, at matched evidence exposure, is ahead of agentic retrievers in sixteen of twenty pairs. The gains persist to 256K-token contexts and at every evidence-hop count. Agentic costs recur with every query. After a break-even number of queries per document, which we report in model calls and in tokens, they exceed SAADI's one-time construction cost. Ablations show that the entity-memory and summary signals each improve retrieval, and that fusing both is best in every setting. Thus document-level connections behind distributed and multi-hop evidence can be established once at index time, at a cost known before the first query. Code and data will be released upon acceptance.

1 INTRODUCTION

Retrieval over a long document has to work with evidence that is spread across the document. A passage may refer to a character only as the *old woman*, while her name is given in another passage, earlier or later in the text. A passage may also be relevant to a question only because of what the surrounding pages are about. In both cases the information that makes the passage relevant is present in the document but not in the chunk itself. A retriever that scores the chunk from its own text alone has no access to it. This has two consequences for an index that is built in one pass over the document, with a predetermined number of model calls, and then used for single-pass retrieval. To be reachable by such a question, a chunk has to be indexed together with what has been established up to that point in the text, namely the entities identified so far and an account of what has been read. And because part of the connecting information appears only later, the entry of an earlier chunk has to be updatable when later text adds to it, without rewriting the chunk. Both requirements concern facts about the document, independent of any query.

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.

Agentic retrieval recovers connections across distant passages at query time, through successive reading guided by the query: by navigating a memory structure (Chen et al., 2023; Lee et al., 2024; Li et al., 2024b), by a learned search policy (Jin et al., 2025; Tan et al., 2025; Liu et al., 2026), or through a structured retrieval interface (Du et al., 2026; Li et al., 2026a). The number of model calls and the evidence read depend on the trajectory, and connections not kept across queries are recovered again. Memory agents build a document memory before any query arrives and consult it during a query-time trajectory (Zhang et al., 2026); this amortizes part of the work, but the number of calls to build the memory depends on the agent's policy, and query-time calls remain. Index-time approaches instead build contextualized chunks (Anthropic, 2024; Günther et al., 2025; Chen et al., 2025), resolve references (Liu et al., 2025a; Xu et al., 2025), or construct summary trees (Sarthi et al., 2024; Liu, 2022) and entity or graph structures (Edge et al., 2024; Gutiérrez et al., 2024; 2025; Zhang et al., 2025a). These representations can encode cross-passage connections, but each is derived from a chunk and its surrounding or global context, not from what was resolved before it, and graph-based ones may also need query-time traversal (Xiang et al., 2026). The challenge is to make these connections available in a single retrieval pass, with no model call at query time, an index-time cost known in advance, and no rebuild when the document grows.

The connections resolved by iterative reading are properties of the document, not of the query. They can be established before any query arrives and reused by every query that follows. Where a document is read many times, it can be understood once. SAADI combines progressive document understanding with a number of model calls that is fixed by the chunk count and one folding factor f, all of them made at index time. When chunks are appended, the state is extended without a rebuild, and the context passed to each call is bounded by a logarithmic number of summaries. The connections established during reading are available to a single retrieval pass over the original chunks, with any base retriever, without graph traversal or text rewriting.

SAADI makes one pass over the chunk sequence at index time. Its state has two parts. An active-entity memory holds the entities identified so far, and a hierarchical history holds an account of the text read so far. Before each chunk is processed, a bounded set of relevant entity records is recalled from the memory and supplied together with a compacted account of the prefix. The memory is then extended with the extracted entities, adding or enriching records and removing none, and the history is folded on a fixed schedule. Each chunk is linked to entity names instead of snapshots of their records, so an alias or description added later reaches every chunk linked to that entity without rewriting it. At query time, two auxiliary channels score each chunk through its linked entity records and its covering summary, and the scores are fused with the base retriever's into a single ranking over the original chunks. The two parts work together; the history conditions extraction at index time, and at query time the linked records and the covering summaries are both effective scoring signals, best when fused (Section 5.3).

We evaluate on four datasets and five retrievers, with a strict hit rate that counts a query only if all of its evidence chunks are in the top-k set. Among fixed-depth methods on the same retriever, SAADI is best or second best in 34 of 40 dataset, retriever, and depth combinations and never below third; the strongest competitor, LQCA (Liu et al., 2025a), which rewrites the indexed text, is far ahead on MuSiQue-32K and QASPER-64K and far behind on NovelHopQA. At comparable evidence exposure, with its depth set to the number of chunks an agent reads, SAADI also outperforms agentic retrieval in 16 of 20 agent-dataset pairs; the four exceptions are on MuSiQue-32K and QASPER-64K. That comparison matches evidence rather than model-call expenditure; SAADI makes no query-time model calls, an agent makes 2.0 to 4.2 per query, and the agent's cumulative calls exceed SAADI's one-time indexing calls after 11-17 queries per document on LooGLE and 279-457 on QASPER-64K. The error analysis places the gain on a subpopulation of queries.

Contributions. **Stateful indexing improves single-pass retrieval of distributed and multi-hop evidence.** We introduce SAADI, which progressively accumulates document state for retrieval over original chunks without query-time model calls, and improves HR@10 over the base retriever in every dataset–retriever pair, by 0.3 to 12.4 points (1.1 to 4.0 under BM25). **A priori calculation of amortization cost.** We derive exact call counts for index construction and chunk appends, a bound on per-call context, and the query volume at which SAADI becomes cheaper than an agentic method, in model calls and in tokens. **Disentangling what the state contributes.** The hierarchical history is what makes conditioned extraction improve the index. At query time the linked records and the covering summaries are both effective scoring signals, and fusing them is best on every dataset.

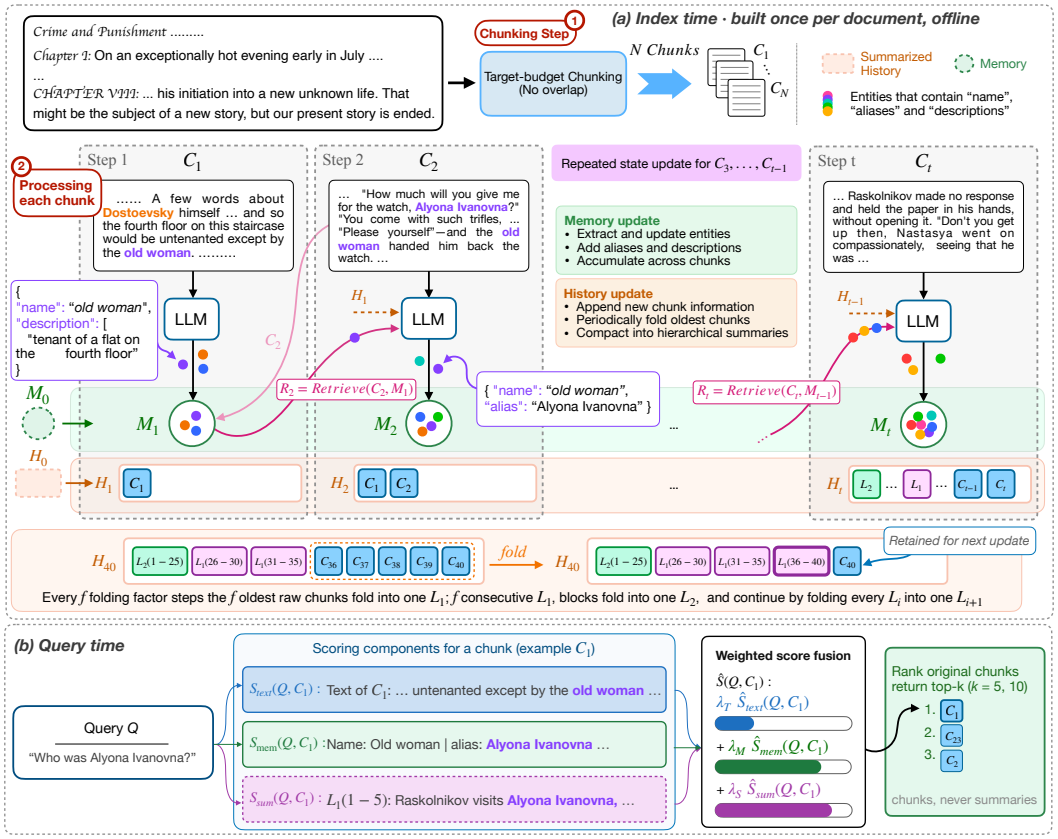

Figure 1: Overview of SAADI. Panel (a) is the offline pass. Each chunk in turn is interpreted by one model call that reads it together with the records recalled from the active-entity memory and the hierarchical history of the prefix, and every f steps the oldest units fold into one summary at each level. Panel (b) is query time, where the text, memory, and summary scores are fused into one ranking over the original chunks, which are never rewritten and are returned without any model call.

2 Related Work

Long-document retrieval methods differ in when a chunk is interpreted and in what that interpretation may see. Appendix A extends this section, gives the comparison axes, and places our all-or-nothing evidence measure among prior criteria.

Document-side structure built before the query. Passage-level retrieval can fragment the evidence a query needs (Dasigi et al., 2021; Lu et al., 2026; Cha et al., 2026). Document-side methods enrich the index offline with contextualized chunk representations (Günther et al., 2025; Anthropic, 2024; Chen et al., 2025), summary trees (Sarthi et al., 2024; Liu, 2022), coreference-normalized text (Liu et al., 2025a; Xu et al., 2025), and entity or graph structures (Zhang et al., 2025a; Edge et al., 2024; Gutiérrez et al., 2024; 2025; Guo et al., 2025). All of them derive a chunk's representation from the chunk and its surrounding or global text in one step; none keeps a state that accumulates as the document is read and is updated by later chunks. Coreference normalization rewrites the indexed text, so the retrieved unit is no longer the original passage. RAPTOR and GraphRAG aggregate globally once every chunk exists, and graph methods may add query-time traversal or filtering (Xiang et al., 2026). Contextual Retrieval also makes one call per chunk, but each call reads the whole document or a window of it; each SAADI call reads a bounded set of recalled records and summaries (Section 3.3).

Conditioning that arrives with the query. A second family waits for the query, expanding or rewriting it (Gao et al., 2023) or interleaving retrieval with reasoning (Trivedi et al., 2023; Jin et al.,

2025). Agentic long-document systems extend this to several rounds of reading, conditioned on previous rounds (Chen et al., 2023; Lee et al., 2024; Zhang et al., 2024b) or on a learned search policy (Tan et al., 2025; Liu et al., 2026), choosing how much of the document to consume (Du et al., 2026; Li et al., 2026a), and sometimes carrying a memory across rounds or queries (Li et al., 2024b; Zhang et al., 2026). They recover the links a single pass misses, but the conditioning is rebuilt for every query, so its cost recurs and the evidence reaching the model varies; Section 5.2 therefore compares at matched evidence exposure. A cross-encoder reranker (Nogueira & Cho, 2019; Nogueira et al., 2020) also conditions on the query but scores a chunk from its own text; it composes with SAADI, which returns original chunks, a combination we do not evaluate.

Conditioning moved to index time. SAADI moves the conditioning to index time. It reads the chunk sequence in order, interpreting each chunk against a selectively recalled active-entity memory and a hierarchical history folded on a fixed schedule. The closest prior construction, SlimRAG (Zhang et al., 2025a), also indexes entities offline, but extracts each chunk's entities from that chunk alone and decomposes the query with a model call at retrieval time, which places it outside our single-pass setting (Appendix A); Appendix C.6 isolates the extraction difference. Because entity signals can have limited coverage (Chatterjee, 2026) and lossy memories can underperform verbatim evidence (An, 2025), the state enters scoring only, and retrieval stays one fixed-depth pass over the original chunks.

3 METHOD

SAADI has two stages (Figure 1). An index-time pass builds a query-agnostic document state with two parts, an active-entity memory and a hierarchical history. At query time, auxiliary scores are derived from this state for a single ranking pass over the original chunks, which are segmented once and never rewritten, merged, or contextualized.

3.1 INDEX-TIME STATE CONSTRUCTION

Setup and state. We write $D = (C_1, \ldots, C_N)$ for the chunks of a document, each formed from consecutive sentences up to a fixed token length (Section 4), without overlap. The state (M_t, H_t), an active-entity memory and a hierarchical history, starts empty. It is updated once per chunk in document order, the order in which the text is written to be read, so C_t is interpreted using only information extracted from earlier chunks. We describe one transition, $(M_{t-1}, H_{t-1}) \to (M_t, H_t)$.

Selective recall. The units of M_{t-1} are active-entity records, each keyed by a descriptive entity *name* and storing its observed *aliases* and short *descriptions*. A small subset $R_t = \mathrm{Retrieve}(C_t, M_{t-1})$ is recalled by scoring C_t against the recorded text with BM25. A fixed budget limits the records recalled and, per record, the aliases and descriptions (Appendices C.1 and C.2). A mention in C_t that shares a surface form with a recorded entity can thus be resolved to it, however early it was introduced.

Hierarchical history. H_{t-1} is a compact record of the document read so far. It holds the recent chunks not yet folded and progressively coarser summaries of older regions. The immediately preceding chunk is always kept verbatim, even when a fold has just absorbed it, because C_t is its direct continuation (Figure 1). Extraction is thus conditioned on what has been established so far, such as the topic of the current section, which is not stated in the chunk itself.

State update. One LLM call then reads the chunk with both parts of the state and returns the active entities the chunk is about,

$$A_t \;=\; \mathrm{LLM}\big(C_t,\, R_t,\, H_{t-1}\big), \qquad M_t \;=\; M_{t-1} \oplus A_t, \tag{1}$$

where $\oplus$ opens a record for a new entity and otherwise extends the existing one with newly observed aliases or descriptions; nothing is removed. Names act as keys. A recalled entity keeps its stored name; a new entity receives a short designation, qualified if it would collide with a recalled record (Appendix B.2). The names returned for C_t are stored as its links into the memory. Because links point to names, not record snapshots, later updates to an entity are available to every chunk linked to

that name; the chunks themselves are not modified. Recall thus precedes extraction; mentions are resolved against earlier entities at processing time, not extracted per chunk and reconciled afterwards. Appendix C.6 varies what extraction may condition on.

Hierarchical folding. The history is then updated with a fixed folding factor f. C_t is appended, and whenever f consecutive units at one level are complete they are replaced by one summary at the level above (Appendix B.3). Each level-one summary thus covers f chunks, and the hierarchy deepens logarithmically in N. Higher levels are used only as extraction context, not for scoring. Scoring uses the level-one summary, of which each chunk has at most one. Coarser levels would add several scores per chunk at different resolutions; how to weight them is left to future work.

What the index stores. The index keeps the unchanged chunks, the memory M_N with the per-chunk name links, and every level-one summary produced during the pass. Every chunk C_i thus maps to the current records of its linked names and, when available, to the level-one summary covering it. Appendix B.4 gives the pass as pseudocode, and Appendix D.1 checks that records and summaries are supported by their source text.

3.2 Query-Time Retrieval

Scoring channels. Given a query Q, every chunk C_i is scored on three channels with the same base retriever, sparse or dense. The text channel $S_{\text{text}}(Q, C_i)$ scores the chunk text and anchors the ranking. The memory channel $S_{\text{mem}}(Q, C_i)$ scores the records linked to C_i field by field (name, aliases, descriptions). The best match over the records is taken per field, and the fields are summed with fixed weights (Appendix C.5.2). The summary channel $S_{\text{sum}}(Q, C_i)$ scores the level-one summary covering C_i. Both auxiliary channels use information that is present in the document but not in the chunk. Through the name links, a connection established anywhere in the document is available when the chunk is scored. A chunk in which a character is named only as *the old woman* does not match, on its own text, a query naming *Alyona Ivanovna*. That alias was added to the linked record when the connection was established in a later chunk, so the memory channel scores the chunk anyway. The summary channel makes a chunk reachable through what its region is about, even when that is not stated in the chunk itself.

Normalization, fusion, and output. The channels are on different scales, and BM25 scores are unbounded, so one term shared with a record or summary can produce a jump that would dominate the sum. Each channel is therefore z-scored over the candidate chunks, written $\hat{S}$, and the channels are combined by a weighted sum,

$$S_{\text{SAADI}}(Q, C_i) \;=\; \lambda_T\, \hat{S}_{\text{text}}(Q, C_i) \;+\; \lambda_M\, \hat{S}_{\text{mem}}(Q, C_i) \;+\; \lambda_S\, \hat{S}_{\text{sum}}(Q, C_i), \tag{2}$$

with weights fixed once for all datasets and retrievers (Section 4), so that the text anchors the ranking and the auxiliary channels adjust it. A chunk with no covering level-one summary receives no contribution from that channel. Appendix C.5.3 compares alternatives. Chunks are ranked by S_{SAADI} and the top k are returned as original text; records and summaries change the order, not the objects ranked. Retrieval is a single pass with no LLM call, no query rewriting, and no adaptive trajectory.

3.3 Construction Cost

The number of construction-time calls is fixed by the chunk count and the folding factor, and the context per call is bounded.

Index-time calls. One state-update call is made per chunk and a fold at level i once per f^i chunks, so the number of index-time model calls for a document of N chunks with f folding factor is

$$T(N, f) \;=\; N + \sum_{i=1}^{\lfloor \log_f N \rfloor} \left\lfloor \frac{N}{f^i} \right\rfloor \;<\; \frac{f}{f-1}\, N, \qquad \frac{T(N, f)}{N} \;\xrightarrow[N \to \infty]{}\; \frac{f}{f-1}. \tag{3}$$

The floors record that incomplete groups are never folded, so the count is exact; at $f = 5$ the hierarchy adds fewer than a quarter of a call per chunk (Appendix B.5.1 proves the bound and the limit). Calls within a document are sequential, since each depends on the state left by the previous one; documents are independent and can be indexed in parallel.

Table 1: Fixed-depth retrieval across four datasets, five retrieval backends, and all compared methods, reported as HR@5 and HR@10 in percent. Dark blue marks the best and light blue the second-best value within each retriever block of a column. Late Chunking requires a long-context encoder and therefore appears under Jina only, at three chunk overlaps.

Retriever	Method	LooGLE		MuSiQue-32K		QASPER-64K		NovelHopQA	
		HR@5	HR@10	HR@5	HR@10	HR@5	HR@10	HR@5	HR@10
BM25	Vanilla	**91.5**	95.4	57.6	74.0	14.4	22.1	**16.9**	**20.0**
	LlamaIndex	91.0	**95.5**	55.4	72.7	**20.4**	**27.0**	3.9	4.5
	LQCA	88.8	94.2	**83.6**	**91.8**	**36.9**	**47.0**	1.4	1.9
	SAADI	**92.3**	**96.5**	**62.8**	**77.7**	17.0	26.1	**18.1**	**22.0**
Contriever	Vanilla	72.4	73.3	59.9	76.1	15.2	23.3	**11.8**	**11.8**
	LlamaIndex	36.3	52.2	50.9	67.7	5.9	11.3	1.9	3.1
	LQCA	**73.3**	**84.5**	**80.8**	**89.0**	**34.4**	**46.0**	0.9	1.7
	SAADI	**75.4**	**85.7**	**68.8**	**83.7**	**16.2**	**25.8**	**13.5**	**18.2**
BGE-M3	Vanilla	**80.6**	**89.8**	73.2	87.4	**21.0**	29.3	**14.1**	**20.7**
	LlamaIndex	80.1	87.3	68.8	82.4	19.2	30.3	13.5	18.2
	LQCA	74.8	84.1	**76.7**	**90.4**	**32.9**	**45.8**	1.4	2.2
	SAADI	**81.1**	**90.1**	**76.9**	**88.3**	20.9	**30.9**	**16.5**	**22.1**
Jina	Vanilla	**70.1**	80.1	71.6	85.8	15.7	24.8	**14.8**	**19.1**
	LateC(No overlap)	63.4	76.8	42.0	59.8	7.9	12.4	**14.8**	17.7
	LateC(Low overlap)	67.5	79.3	41.7	58.1	7.9	11.1	14.3	17.7
	LateC(More overlap)	67.7	80.3	36.2	49.4	7.4	10.3	**14.6**	17.3
	LlamaIndex	**71.6**	**81.5**	67.3	81.9	**21.2**	**29.2**	3.1	4.2
	LQCA	68.6	79.1	**79.4**	**89.0**	**31.7**	**41.8**	2.1	2.2
	SAADI	**70.1**	**82.0**	**73.2**	**88.1**	18.9	27.2	**14.8**	**21.1**
Qwen	Vanilla	76.9	85.9	75.3	89.0	22.6	32.6	**14.6**	**19.8**
	LlamaIndex	50.8	57.3	52.4	58.1	22.6	28.6	2.4	2.8
	LQCA	**81.4**	**86.8**	**82.2**	**90.4**	**39.7**	**53.6**	2.2	2.8
	SAADI	**77.0**	**86.7**	**76.4**	**90.0**	**23.3**	**34.1**	**15.2**	**20.6**

Per-call context and extension. Each call reads the current chunk, a bounded set of recalled records, and the history. After t chunks the history holds at most $(f-1)(\lfloor \log_f t \rfloor + 1)$ units, each a chunk or a summary of bounded length, plus one retained chunk (Appendix B.3), so the context per call grows logarithmically with the document. The pass never revisits a processed chunk, so a growing document is extended, not rebuilt. Appending C_{N+1} costs one state-update call plus one fold per power of f dividing $N+1$, at most $1+\lfloor \log_f(N+1) \rfloor$ calls and $f/(f-1)$ amortized, 1.25 at $f=5$ (Appendix B.5.2).

Break-even against per-query computation. SAADI's cumulative call count stays at $T(N,f)$ however many queries are issued against a document. An alternative method with no index-time cost that averages $\bar{c}$ calls per query reaches the same total after

$$Q^\star = \frac{T(N,f)}{\bar{c}} \approx \frac{f}{f-1} \cdot \frac{N}{\bar{c}} \tag{4}$$

queries. Only $\bar{c}$ is measured; Section 5 reports it by method and dataset. The same accounting covers token usage and methods that also pay an index-time cost (Appendix B.5.3).

4 Experimental Setup

Data, chunking, and conditions. We evaluate on LooGLE (Li et al., 2024a), MuSiQue (Trivedi et al., 2022), QASPER (Dasigi et al., 2021), and NovelHopQA (Gupta et al., 2025). Documents are segmented at sentence boundaries into non-overlapping chunks of 100 tokens for MuSiQue and QASPER and 500 tokens for LooGLE and NovelHopQA, following prior work (Bhat et al., 2025; Bennani & Moslonka, 2026). LooGLE and NovelHopQA are used in their native form. MuSiQue and QASPER are expanded to 32K, 64K, 128K, and 256K tokens by concatenating documents of the same corpus, each the evidence for its own queries and a distractor for the others (Yang et al., 2025b; Yuan et al., 2024); the main results use MuSiQue-32K and QASPER-64K. Query populations and the expansion protocol are given in Appendix E.1.

Evidence units and hit rate. Annotated evidence is mapped onto the fixed chunks, so a query's target is the set of chunks its evidence occupies, several when a span crosses a boundary or the hops

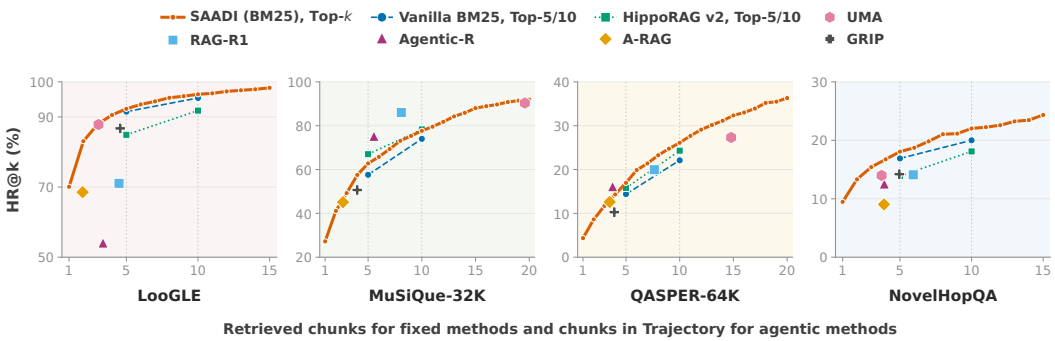

Figure 2: Hit rate against the evidence available before answer generation. Fixed-depth methods are placed at their cutoff and agentic methods at the mean number of distinct chunks that appeared in their trajectory content that enter a model call before the answer. For A-RAG, this counts full-chunk reads and excludes search-result previews.

of a question lie in separate passages. A query counts at depth k only when its whole target is in the top k; HR@k is the fraction of queries retrieved, at $k \in \{5, 10\}$. A target larger than k cannot be retrieved at that depth, so the measure is strict (Appendix E.3).

SAADI configuration. The state is built once per document at folding factor $f = 5$ with `Qwen3-30B-A3B-Instruct-2507` (Yang et al., 2025a), with a recall budget of five records, two aliases, and three descriptions. The channel weights are $(\lambda_T, \lambda_M, \lambda_S) = (1, 0.5, 0.3)$ and the field weights $(\mu_N, \mu_A, \mu_D) = (1, 1, 1)$ under z-score normalization, selected once on 100 random queries from datasets LooGLE and NovelHopQA and held fixed for every dataset, retriever, depth, and condition (Appendix E.5).

Retrievers and baselines. Query-time scoring is evaluated with BM25 (Robertson & Zaragoza, 2009), Contriever (Izacard et al., 2021), `jina-embeddings-v3` (Sturua et al., 2024), `Qwen3-Embedding-8B` (Zhang et al., 2025b), and `bge-m3` (Chen et al., 2024), with BM25 for ablations. Fixed-depth baselines are Vanilla retrieval, Late Chunking (Günther et al., 2025), LLAMAINDEX (Liu, 2022), LQCA (Liu et al., 2025a), HippoRAG v2 (Gutiérrez et al., 2025) in its default configuration, and RAPTOR (Sarthi et al., 2024), all at the same k and, within a dataset, on the same query population; RAPTOR and Late Chunking are scored under conventions in their favour (Appendix E.6). Agentic baselines, RAG-R1 (Tan et al., 2025), Agentic-R (Liu et al., 2026), A-RAG (Du et al., 2026), GRIP (Li et al., 2026a), and UMA (Zhang et al., 2026), choose how much to read, so they are compared by pre-answer chunk exposure, the distinct chunks whose content enters a model call before the answer, against SAADI under BM25 at depths 3 to 20. Each agent is run as released, with its policy model, prompts, and search tool unchanged, over the same chunk sequence; we only record its trajectory, cut at the start of answer generation.

5 RESULTS

5.1 RETRIEVAL EFFECTIVENESS AND RANKING STABILITY

Table 1 gives HR@5 and HR@10 for the fixed-depth methods on four datasets under five retrievers. SAADI is above its base retriever at HR@10 in all twenty dataset-retriever pairs, with gains between 0.3 and 12.4 points. Under BM25 the gain is 1.1 points on LooGLE, 3.7 on MuSiQue-32K, 4.0 on QASPER-64K, and 2.0 on NovelHopQA. LooGLE leaves little room, since a document there holds about 32 chunks and BM25 alone already returns the full target for 95.4 percent of queries at depth 10. Even that gain is more than three times the run-to-run standard deviation (Appendix E.5).

Among fixed-depth methods on the same retriever, SAADI is first or second in 34 of the 40 dataset, retriever, and depth cells and never below third (Appendix F.1). The count measures consistency, not dominance. LQCA, which rewrites the indexed text, is far ahead on QASPER-64K, 47.0 against 26.1 HR@10 under BM25, and far behind on NovelHopQA, at 1.7 to 2.8 under every retriever. HippoRAG v2, outside the grid, is below SAADI at depth 10 on three of four datasets and above

Table 2: Agentic retrieval against SAADI at matched evidence exposure. For each agent, the trajectory columns give the mean number of chunks it reads before answering ($\bar{k}$) and its hit rate over those chunks; the SAADI columns give SAADI's hit rate under BM25 at the two integer top-k bracketing $\bar{k}$, bold where it exceeds the agent's (full curve at Figure 2). The last two columns give the agent's pre-answer model calls per query (SAADI makes none) and the number of queries per document at which those calls add up to SAADI's one-time construction cost, with UMA's own offline memory construction subtracted first.

Dataset	Agent	Agent Trajectory		SAADI at Matched Depth				LLM Calls	
		Avg. Chunks($\bar{k}$)	HR	$k{=}\lfloor\bar{k}\rfloor$	HR@k	$k{=}\lceil\bar{k}\rceil$	HR@k	Per Query (Agent)	SAADI Crossover (Queries/Doc.)
LooGLE	**RAG-R1**	4.49	71.07	4	**90.56**	5	**92.30**	2.43	16
	Agentic-R	3.37	54.09	3	**87.94**	4	**90.56**	2.65	15
	A-RAG	1.95	68.56	1	**70.09**	2	**83.08**	2.25	17
	GRIP	4.57	86.74	4	**90.56**	5	**92.30**	2.71	14
	UMA	3.06	87.83	3	**87.94**	4	**90.56**	2.01	11
MuSiQue-32K	**RAG-R1**	8.09	86.00	8	73.11	9	75.33	4.15	114
	Agentic-R	5.54	75.22	5	62.78	6	65.78	3.70	128
	A-RAG	2.66	45.11	2	41.11	3	**49.22**	3.31	143
	GRIP	3.99	50.67	3	49.22	4	**57.56**	2.46	192
	UMA	19.59	90.33	19	**91.44**	20	**91.89**	2.71	164
QASPER-64K	**RAG-R1**	7.64	19.95	7	**21.38**	8	**23.26**	3.34	279
	Agentic-R	3.76	16.13	3	11.70	4	14.33	2.71	344
	A-RAG	3.48	12.60	3	11.70	4	**14.33**	2.03	457
	GRIP	3.91	10.28	3	**11.70**	4	**14.33**	2.40	388
	UMA	14.79	27.31	14	**31.13**	15	**32.33**	2.11	413
NovelHopQA	**RAG-R1**	5.93	14.10	5	**18.06**	6	**18.72**	2.42	221
	Agentic-R	3.91	12.56	3	**15.42**	4	**16.74**	2.67	201
	A-RAG	3.89	9.03	3	**15.42**	4	**16.74**	2.09	256
	GRIP	4.95	14.21	4	**16.74**	5	**18.06**	2.60	206
	UMA	3.72	13.99	3	**15.42**	4	**16.74**	2.19	169

on MuSiQue-32K, 78.3 against 77.7. When the retriever is chosen on the other three datasets only, SAADI keeps the best held-out mean rank, 1.50, never ranks below second, and has the smallest worst-case regret, 6.4 points (Appendix F.2).

5.2 Agentic retrieval, effectiveness and cost

An agent decides how much of a document to read, so its evidence varies by query and a model call is not a fixed amount of evidence; we therefore compare at matched evidence. Each agent's mean pre-answer exposure, the distinct chunks whose content enters a model call before the answer, is set against SAADI under BM25 at the two integer depths bracketing it (Table 2; full curves in Figure 2). SAADI is ahead at the nearest depth in 16 of the 20 agent and dataset pairs, and at both bracketing depths in 14. On LooGLE and NovelHopQA, it is ahead of every agent at both depths. The four exceptions are RAG-R1 and Agentic-R on MuSiQue-32K and Agentic-R and A-RAG on QASPER-64K, the two corpora where the SAADI curve rises slowly over the first few depths. No agent keeps its rank across corpora. GRIP is the strongest agent on NovelHopQA and the weakest on QASPER-64K, and UMA, the strongest on the other three, reads from 3.1 chunks per query on LooGLE to 19.6 on MuSiQue-32K (Appendix F.4).

The ordering is stable as documents grow and as queries need more hops (Figure 3). Every method loses accuracy as distractors are added, but SAADI keeps its margin over its base retriever at every length up to 256K tokens on MuSiQue and QASPER, under BM25 and Contriever, and keeps its place against the agents. On QASPER, the best depth-10 row at every length is a SAADI row, above the four agents that read up to 9.5 chunks; UMA, which reads about fifteen, is above it at three of the five lengths. On MuSiQue, agent exposure grows while SAADI's depth does not, and RAG-R1, at 6.85 to 9.45 chunks, leads at every expanded target (Appendix F.6). Across hop counts, the same holds. SAADI is ahead of its base retriever at every hop count on both multi-hop datasets, most narrowly at four hops on NovelHopQA, 11.0 against 10.5. On NovelHopQA, the best cell at every hop count is a SAADI row, with every agent below; on MuSiQue-32K the agents gain with hops (Appendix F.3).

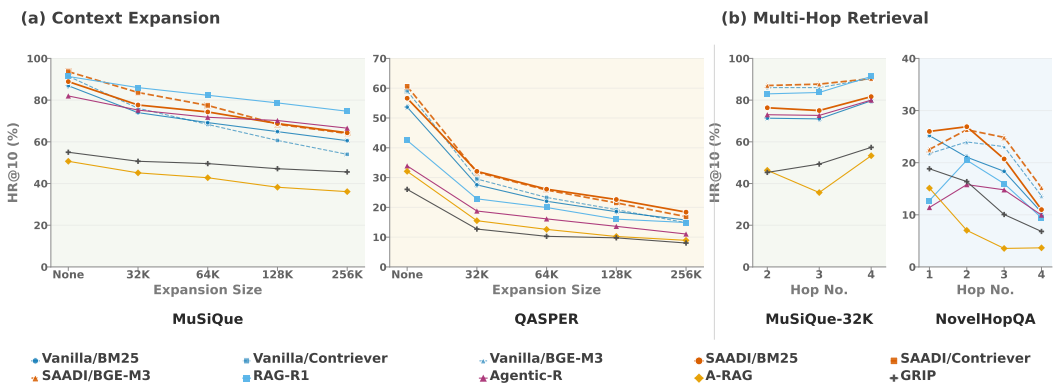

Figure 3: Retrieval under increasing long-document difficulty. Panel (a) evaluates context expansion, and panel (b) stratifies retrieval by the number of evidence hops. Both panels report HR@10.

SAADI makes no model call at query time, and its construction cost is fixed by the chunk count and the folding factor (Section 3.3). An agent makes 2.0 to 4.2 calls per query (Table 2). Equation (4) turns these into the number of queries per document at which an agent's cumulative calls reach SAADI's construction cost, with UMA's own offline memory construction subtracted. Table 2 places this crossover at 11 to 17 queries on LooGLE and 279 to 457 on QASPER-64K; longer documents cost more to build and take longer to repay (Appendices F.4 and F.5). Measured in generated tokens, which construction spends on writing records and summaries, the crossovers are about an order of magnitude larger (Appendix F.4). A deployment expecting fewer queries per document than the crossover is cheaper served by the agent, and one expecting more by SAADI.

5.3 Ablations

We first test how the state should be constructed by rebuilding the index under four variants: chunk-only extraction, history-conditioned extraction, memory-conditioned extraction, and full history+memory conditioning. Query-time scoring is kept identical across all variants, so differences isolate the effect of state construction (Table 27, Appendix F.7). At index time the gain comes from the history. Conditioning extraction on it improves over unconditioned extraction on every dataset, by up to 3.0 points on MuSiQue-32K. Conditioning on recalled records alone changes HR@10 by no more than 0.4 points, and adding the records to the history moves it by -0.9 to $+0.3$. At query time both auxiliary channels help on every dataset, and fusing them is best everywhere. The summary channel is the stronger single signal on MuSiQue-32K and NovelHopQA, and the two channels are within 0.2 points of each other on LooGLE and QASPER-64K. The state therefore helps in two different places, the history at extraction time and the entity memory as a scoring signal. The weights are fixed, so two auxiliary channels that each keep a chunk inside the top 10 can together push it out (Appendix F.9). And the construction model matters less than the retriever: replacing the 30B model with a 7B, 24B, or 70B alternative moves HR@10 by at most 1.4 points (Appendix C.3).

6 Conclusion

SAADI builds up document understanding in a single index-time pass and makes it available to single-pass retrieval over the original chunks, with an analytically determined number of model calls for index construction and extension and none at query time. Given the expected per-query usage of an agent, we can compute the query volume at which SAADI's index-time investment amortizes. Across the evaluated datasets and retrievers, SAADI delivers consistently competitive retrieval, outperforming agents at comparable evidence exposure in sixteen of the twenty dataset-agent pairs, and its improvements over the base retriever persist across context expansions and evidence-hop counts. SAADI enables multi-hop evidence retrieval in a single retrieval pass through constructing an index from iterative accumulation of document state. Whether this index construction is worthwhile can be estimated before construction based on the document set and the expected per-query cost of agentic alternatives.

Reproducibility Statement

The method is specified in Section 3 and Appendix B, which give the pseudocode, the folding schedule, the prompts and output contracts, and the derivations of the cost results. Section 4 and Appendix E give the datasets, the chunking, the query populations, the expansion protocol, the retriever and construction checkpoints, the decoding settings, the fusion weights and how they were selected, and the accounting rules applied to every baseline. Appendix E.5 reports three independent constructions and the resulting variation. Code and data will be available upon acceptance.

AI Use Statement

We used general-purpose assistants to edit the text for clarity, to search for related work, and to review and improve our code, and every reference was checked against the original publication. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

Ghadir Alselwi, Basem Suleiman, Hao Xue, Shoaib Jameel, Hakim Hacid, Flora D Salim, and Imran Razzak. Knowledge graph enhanced memory-augmented retrieval for long context modeling. *arXiv preprint arXiv:2606.14047*, 2026.

Tao An. Verbatim chunks beat extracted artifacts: A controlled ablation of memory representations for long llm conversations. *arXiv preprint arXiv:2601.00821*, 2025.

Anthropic. Introducing contextual retrieval. `https://www.anthropic.com/news/contextual-retrieval`, September 2024. Accessed 2026.

Mohsen Arjmandi. Distractor-aware truncation: Disentangling context-length effects from signal loss in long-context llm benchmarks. *arXiv preprint arXiv:2608.03297*, 2026.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pp. 9112–9141, 2024.

Sofia Bennani and Charles Moslonka. A systematic analysis of chunking strategies for reliable question answering. In *European Conference on Information Retrieval*, pp. 68–73. Springer, 2026.

Sinchana Ramakanth Bhat, Max Rudat, Jannis Spiekermann, and Nicolas Flores-Herr. Rethinking chunk size for long-document retrieval: A multi-dataset analysis. *arXiv preprint arXiv:2505.21700*, 2025.

Sungguk Cha, DongWook Kim, Mintae Kim, Youngsub Han, Byoung-Ki Jeon, and Sangyeob Lee. Do current retrievers cover all the evidence? a controlled study of conjunctive cross-page retrieval. *arXiv preprint arXiv:2607.24165*, 2026.

Shubham Chatterjee. Entities as retrieval signals: A systematic study of coverage, supervision, and evaluation in entity-oriented ranking. *arXiv preprint arXiv:2604.05204*, 2026.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*, 2023.

Huiyao Chen, Yi Yang, Yinghui Li, Meishan Zhang, Baotian Hu, and Min Zhang. Beyond chunking: Discourse-aware hierarchical retrieval for long document question answering. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18176–18198, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.829. URL `https://aclanthology.org/2026.acl-long.829/`.

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2318–2335, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.137. URL https://aclanthology.org/2024.findings-acl.137/.

Peter Baile Chen, Tomer Wolfson, Mike Cafarella, and Dan Roth. Enrichindex: Using LLMs to enrich retrieval indices offline. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=wyYL5Jov6e.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 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, 2009.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 719–729, 2024.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4599–4610, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.365. URL https://aclanthology.org/2021.naacl-main.365/.

Trishita Dhara and Siddhesh Sheth. Context-length robustness in question answering models: A comparative empirical study. *arXiv preprint arXiv:2603.15723*, 2026.

Kuicai Dong, Yujing Chang, Derrick Goh Xin Deik, Dexun Li, Ruiming Tang, and Yong Liu. MMDocIR: Benchmarking multimodal retrieval for long documents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 30971–31005, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1576. URL https://aclanthology.org/2025.emnlp-main.1576/.

Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. A-rag: Scaling agentic retrieval-augmented generation via hierarchical retrieval interfaces. *arXiv preprint arXiv:2602.03442*, 2026.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Edward A Fox and Joseph A Shaw. Combination of multiple searches. *NIST special publication SP*, 243, 1994.

Jorge Gabín, Anxo Perez, and Javier Parapar. Lost in the evidence? reproducing document position and context size effects in rag. *arXiv preprint arXiv:2605.27105*, 2026.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.99. URL https://aclanthology.org/2023.acl-long.99/.

Yunfan Gao, Yun Xiong, Wenlong Wu, Bohan Li, Yijie Zhong, and Haofen Wang. U-niah: Unified rag and llm evaluation for long context needle-in-a-haystack. *ACM Transactions on Information Systems*, 44(3):1–30, 2026.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. Late chunking: Contextual chunk embeddings using long-context embedding models, 2025. URL https://openreview.net/forum?id=74QmBTV0Zf.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. Lightrag: Simple and fast retrieval-augmented generation. In *EMNLP (Findings)*, pp. 10746–10761, 2025.

Abhay Gupta, Kevin Zhu, Vasu Sharma, Sean O'Brien, and Michael Lu. NovelHopQA: Diagnosing multi-hop reasoning failures in long narrative contexts. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 26134–26151, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1328. URL https://aclanthology.org/2025.emnlp-main.1328/.

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

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Omar Khattab, Christopher Potts, and Matei Zaharia. Baleen: Robust multi-hop reasoning at scale via condensed retrieval. *Advances in Neural Information Processing Systems*, 34:27670–27682, 2021.

Seemab Latif, Huma Ameer, Muhammad Hannan Akram, and Mehwish Fatima. The chunking paradigm: Recursive semantic for RAG optimization. In Mourad Abbas, Tariq Yousef, and Lukas Galke (eds.), *Proceedings of the 8th International Conference on Natural Language and Speech Processing (ICNLSP-2025)*, pp. 137–145, Southern Denmark University, Odense, Denmark, August 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.icnlsp-1.15/.

Kuang-Huei Lee, Xinyun Chen, Hiroki Furuta, John Canny, and Ian Fischer. A human-inspired reading agent with gist memory of very long contexts. *arXiv preprint arXiv:2402.09727*, 2024.

Sangam Lee, Ryang Heo, SeongKu Kang, and Dongha Lee. Imagine all the relevance: Scenario-profiled indexing with knowledge expansion for dense retrieval. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ZYVAtUUNbH.

Quinn Leng, Jacob Portes, Sam Havens, Matei Zaharia, and Michael Carbin. Long context rag performance of large language models. *arXiv preprint arXiv:2411.03538*, 2024.

Bo Li, Mingda Wang, Gexiang Fang, Shikun Zhang, and Wei Ye. Retrieval as generation: A unified framework with self-triggered information planning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4254–4274, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.196. URL https://aclanthology.org/2026.acl-long.196/.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. LooGLE: Can long-context language models understand long contexts? In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16304–16333, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.859. URL https://aclanthology.org/2024.acl-long.859/.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12758–12786, 2024b.

Zhanli Li, Huiwen Tian, Lvzhou Luo, Yixuan Cao, and Ping Luo. Deepread: Document structure-aware reasoning to enhance agentic search. *arXiv preprint arXiv:2602.05014*, 2026b.

Jerry Liu. Llamaindex, November 2022. URL https://github.com/run-llama/llama_index.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. Agentic-r: Learning to retrieve for agentic search. *arXiv preprint arXiv:2601.11888*, 2026.

Yanming Liu, Xinyue Peng, Jiannan Cao, Yanxin Shen, Tianyu Du, Sheng Cheng, Xun Wang, Jianwei Yin, and Xuhong Zhang. Bridging context gaps: Leveraging coreference resolution for long contextual understanding. In *International Conference on Learning Representations*, volume 2025, pp. 64318–64334, 2025a.

Yu-An Liu, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. Robust neural information retrieval: An adversarial and out-of-distribution perspective. *ACM Transactions on Information Systems*, 44(1):1–48, 2025b.

Wensheng Lu, Keyu Chen, Zhifeng Shen, Ruizhi Qiao, and Xing Sun. HiChunk: Evaluating and enhancing retrieval augmented generation with hierarchical chunking. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29738–29753, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1372. URL https://aclanthology.org/2026.acl-long.1372/.

Shanshan Lyu, Yiwei Wang, Yujun Cai, Jiafeng Guo, and Shenghua Liu. Lost in a single vector: Improving long-document retrieval with chunk evidence aggregation. *arXiv preprint arXiv:2606.18781*, 2026.

Andrianos Michail, Elias Schuhmacher, Juri Opitz, Simon Clematide, and Rico Sennrich. Attention calibration for position-fair dense information retrieval. *arXiv preprint arXiv:2606.02737*, 2026.

Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan A. Rossi, Seunghyun Yoon, and Hinrich Schuetze. Nolima: Long-context evaluation beyond literal matching. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=0OshX1hiSa.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*, 2019.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pre-trained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL https://aclanthology.org/2020.findings-emnlp.63/.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.378. URL https://aclanthology.org/2023.findings-emnlp.378/.

Hongjin Qian, Zheng Liu, Peitian Zhang, Kelong Mao, Defu Lian, Zhicheng Dou, and Tiejun Huang. Memorag: Boosting long context processing with global memory-enhanced retrieval augmentation. In *Proceedings of the ACM on Web Conference 2025*, pp. 2366–2377, 2025.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. Zep: A temporal knowledge graph architecture for agent memory, 2025. *URL https://arxiv.org/abs/2501.13956*, 2025.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and trends® in information retrieval*, 4(1-2):1–174, 2009.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pp. 42–49, 2004.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher Manning. Raptor: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations*, volume 2024, pp. 32628–32649, 2024.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9248–9274, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.620. URL https://aclanthology.org/2023.findings-emnlp.620/.

Muhammad Arslan Shaukat, Muntasir Adnan, and Carlos CN Kuhn. A systematic investigation of document chunking strategies and embedding sensitivity. *arXiv preprint arXiv:2603.06976*, 2026.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, et al. jina-embeddings-v3: Multilingual embeddings with task lora. *arXiv preprint arXiv:2409.10173*, 2024.

Simeng Sun, Yang Liu, Shuohang Wang, Dan Iter, Chenguang Zhu, and Mohit Iyyer. Pearl: Prompting large language models to plan and execute actions over long documents. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 469–486, 2024.

Zhiwen Tan, Jiaming Huang, Qintong Wu, Hongxuan Zhang, Chenyi Zhuang, and Jinjie Gu. Rag-r1: Incentivizing the search and reasoning capabilities of llms through multi-query parallelism. *arXiv preprint arXiv:2507.02962*, 2025.

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

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Jeff Wu, Long Ouyang, Daniel M Ziegler, Nisan Stiennon, Ryan Lowe, Jan Leike, and Paul Christiano. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Zhishang Xiang, Chuanjie Wu, Qinggang Zhang, Shengyuan Chen, Zijin Hong, Xiao Huang, and Jinsong Su. When to use graphs in rag: A comprehensive analysis for graph retrieval-augmented generation. In *International Conference on Learning Representations*, volume 2026, pp. 66145–66178, 2026.

Huanwei Xu, Lin Xu, and Liang Yuan. Clap: Coreference-linked augmentation for passage retrieval. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 3612–3622, 2025.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

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

Wang Yang, Hongye Jin, Shaochen Zhong, Song Jiang, Qifan Wang, Vipin Chaudhary, and Xiaotian Han. 100-longbench: Are de facto long-context benchmarks literally evaluating long-context ability? *arXiv preprint arXiv:2505.19293*, 2025b.

Daegon Yu, SeungYoon Han, and Woomyoung Park. Is position bias in dense retrievers built in-or learned from data? *arXiv preprint arXiv:2605.26578*, 2026.

Hongli Yu, Tinghong Chen, Jiangtao Feng, Jiangjie Chen, Weinan Dai, Qiying Yu, Ya-Qin Zhang, Wei-Ying Ma, Jingjing Liu, Mingxuan Wang, et al. Memagent: Reshaping long-context llm with multi-conv rl-based memory agent, 2025. *URL https://arxiv. org/abs/2507.02259*, 2259, 2025.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. LV-Eval: A balanced long-context benchmark with 5 length levels up to 256k. *arXiv preprint arXiv:2402.05136*, 2024.

Ziyang Zeng, Dun Zhang, Jiacheng Li, Zoupanxiang, Yudong Zhou, and Yuqing Yang. An empirical study of position bias in modern information retrieval. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5069–5081, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.271. URL https://aclanthology.org/2025.findings-emnlp.271/.

Jiale Zhang, Jiaxiang Chen, Zhucong Li, Jie Ding, Kui Zhao, Zenglin Xu, Xin Pang, and Yinghui Xu. Slimrag: Retrieval without graphs via entity-aware context selection. *arXiv preprint arXiv:2506.17288*, 2025a.

Kehao Zhang, Shangtong Gui, Sheng Yang, Wei Chen, and Yang Feng. Learning to remember: End-to-end training of memory agents for long-context reasoning. *arXiv preprint arXiv:2602.18493*, 2026.

Weinan Zhang, Junwei Liao, Ning Li, Kounianhua Du, and Jianghao Lin. Agentic information retrieval. *arXiv preprint arXiv:2410.09713*, 2024a.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025b.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arık. Chain of agents: Large language models collaborating on long-context tasks. *Advances in Neural Information Processing Systems*, 37:132208–132237, 2024b.

Pengcheng Zhou, Haochen Li, Zhiqiang Nie, JiaLe Chen, Qing Gong, Weizhen Zhang, and Chun Yu. Understand then memory: A cognitive gist-driven rag framework with global semantic diffusion. *arXiv preprint arXiv:2602.15895*, 2026.

Dawei Zhu, Liang Wang, Nan Yang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. LongEmbed: Extending embedding models for long context retrieval. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 802–816, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.47. URL https://aclanthology.org/2024.emnlp-main.47/.

A Extended Related Work and Comparison Scope

This appendix expands Section 2. It first defines the retrieval interfaces and accounting used in our comparisons, while the experimental procedures themselves are recorded in Appendix E. The remaining sections cover document-side construction, sequential memory, query-time retrieval, the measurement of distributed evidence, composability, and robustness.

A.1 Scope of the Comparison and Retrieval Interfaces

Long-document retrieval methods differ in what they return, when they use additional model computation, and how much of a document they expose before producing an answer. We define four axes so that the reported comparisons have a common interpretation. The axes decide which comparisons are well posed, while Appendix E.6 records how each baseline was actually run.

Where additional model computation is spent. We distinguish computation performed during index construction from computation repeated for each query. This accounting covers processing beyond the base retriever used to score candidate passages. It determines whether a cost is amortized across queries or incurred again for every query, as examined in Section 3.3.

What retrieval returns. A retrieval unit may be an original passage, rewritten or normalized text, a generated summary, or a node in a tree or graph. Passage-level hit rates are defined directly when original passages are returned. Other units require a mapping back to the passages covered by the retrieved representation.

Whether retrieval depth is fixed. A fixed-depth method returns k units under a budget chosen in advance. An adaptive method decides how much to retrieve after observing the query. No single k describes the evidence consumed by such a method.

What is counted. For a fixed-depth method, the returned set is both the evidence exposed to the reader and the set being scored. Iterative methods may expose evidence over several retrieval steps. Scoring only their final selection would omit chunks that entered the trajectory and could influence later decisions. The rule this implies for the agentic systems, and the depths at which SAADI is compared against them, are stated in Appendix E.6.

Table 3 places the evaluated configurations on these axes, together with the two closest alternatives that are not run.

Table 3: Placement of the evaluated configurations on the four comparison axes, with the two closest alternatives that are not run below the rule. Additional model computation excludes the base retriever that scores candidate passages, and Accounting is the set over which retrieval success is measured.

Method	Additional model computation	Retrieval unit	Depth	Accounting
Vanilla	none	original passage	fixed	returned set
Late Chunking	index-time encoder	original passage	fixed	returned set
LLAMAINDEX	index time	passage or summary node	fixed	returned set
LQCA	index-time coreference	mention-replaced passage	fixed	returned set
RAPTOR	index-time generation	summary or leaf node	fixed in evaluation	descendant passage set
HippoRAG v2	index-time extraction, query-time filtering	original passage	fixed	returned set
SLIMRAG	index- and query-time LLM	original passage	fixed	returned set
Agentic systems	query-time LLM	passage or structured node	adaptive	exposed set
UMA	training, index, and query time	memory and passage	adaptive	exposed set
SAADI	index-time LLM	original passage	fixed	returned set
Contextual Retrieval (not run)	index-time LLM, whole-document or windowed input	passage with prepended context	fixed	returned set
Cross-encoder reranker (not run)	query-time encoder	original passage	fixed	returned set

RAPTOR and the upper bound. RAPTOR returns nodes of a summary tree, so the second axis separates it from the methods that return passages. Scoring it at a chunk budget needs a mapping chosen by the evaluator rather than by the method, and we adopt the most favourable one, which makes its rows a ceiling rather than an operating point. Appendix E.6 states the conversion.

HippoRAG v2 at query time. HippoRAG v2 filters candidate triples with a model call at query time in its published pipeline, and we run it in that default configuration. It stays in the fixed-depth comparison because it returns original passages at a fixed depth, but it does not share SAADI's property of making no model call at query time.

Methods outside the main comparison. SLIMRAG returns original passages at a fixed depth, but it also invokes a model at query time to decompose the query and extract entities from the parts. The decomposition changes the query, so its retrieval is no longer a single pass over a fixed query, and it falls outside the setting isolated by the main comparison. The document-side difference most relevant to SAADI is whether entity extraction is performed independently for each chunk or conditioned on accumulated document state. Appendix C.6 tests this distinction while holding retrieval and fusion fixed. This ablation isolates the construction difference and does not reproduce SlimRAG's query-time pipeline.

Two alternatives not run. Contextual Retrieval is the closest one-call-per-chunk construction. Its per-call input is the whole document or a window of it, whereas each SAADI call reads a bounded set of recalled records and summaries (Section 3.3), so the two occupy different cost regimes as documents grow; a comparison at equal budgets is left open in Section 6. A cross-encoder reranker adds no index-time cost and scores each candidate from its own text, so information absent from the chunk stays absent. It composes with SAADI, which returns original chunks (Appendix A.7), and the stacked comparison is likewise left open.

Why retrieval rather than long-context reading. A long-context model may receive an entire document instead of a retrieved subset. This option does not remove the retrieval problem when the document exceeds the available window. Reading quality within the window also remains sensitive to evidence position and lexical overlap (Liu et al., 2024; Modarressi et al., 2025; Leng et al., 2024; Dhara & Sheth, 2026). Our evaluation measures which evidence is selected for the reader rather than how the reader uses that evidence.

A.2 LONG-DOCUMENT RETRIEVAL AND CHUNKING

Long-document retrieval commonly represents a document as independently retrievable passages. This can separate evidence needed for the same information need across passage boundaries. QASPER reports that 55.5% of its answerable text-evidence questions require evidence from multiple paragraphs (Dasigi et al., 2021). HiCBench distinguishes questions whose evidence remains within one semantic chunk from those whose evidence spans chunks (Lu et al., 2026). Conjunctive queries over multi-page documents similarly reveal a gap between retrieving partial and complete evidence (Cha et al., 2026). Long-document benchmarks have also introduced graded relevance and page-level or layout-level retrieval (Zhu et al., 2024; Dong et al., 2025). Work on single-vector

representations identifies a related loss of local information when a long document is compressed before the query is known (Lyu et al., 2026).

One response is to modify segmentation or chunk representation. Retrieval quality is sensitive to chunk size and segmentation strategy (Bhat et al., 2025; Shaukat et al., 2026; Bennani & Moslonka, 2026). Other methods derive semantic or hierarchical boundaries (Chen et al., 2026; Latif et al., 2025). Practical frameworks such as LLAMAINDEX support hierarchical and window-based indexing variants (Liu, 2022). LATE CHUNKING encodes a longer document context before pooling chunk representations, allowing each embedding to retain information from surrounding text (Günther et al., 2025). Contextual Retrieval instead attaches generated document-specific context to chunks before indexing (Anthropic, 2024), in the way document expansion attaches predicted queries (Nogueira et al., 2019). Both read context that surrounds a chunk or spans the whole document, without an order in which earlier chunks are resolved before later ones, which is the distinction drawn in Section 2. An alias introduced some pages earlier reaches a summary or a late-chunked embedding only as neighbouring text, and a contextualized chunk only if every call reads the whole document. Contextual Retrieval's per-call input therefore grows with the document or must be windowed, whereas each SAADI call reads a bounded set of recalled records and summaries. They are otherwise compatible with the retrieval setting studied here. SAADI holds segmentation and passage text fixed so that the effect of accumulated document state can be measured without changing the evidence units.

A.3 INDEX-TIME STRUCTURE AND ENTITY SIGNALS

Index-time methods enrich retrieval through offline-generated features (Chen et al., 2025; Lee et al., 2025), coreference normalization, hierarchical summaries, and entity structure. LQCA performs long-context coreference resolution and replaces mentions with representative expressions, making cross-context references easier to recover while changing the indexed text (Liu et al., 2025a). RAPTOR recursively clusters and summarizes passages into a hierarchy and retrieves across levels of abstraction (Sarthi et al., 2024), following earlier recursive summarization of long texts (Wu et al., 2021). SAADI uses its hierarchy differently. The hierarchy maintains a compact history that conditions later extraction, and its level-one summaries also score the original chunks as one auxiliary channel; the hierarchy itself is never returned as evidence. GraphRAG builds a community structure over extracted entities and relations for query-focused summarization (Edge et al., 2024), LightRAG couples an entity–relation graph with vector retrieval (Guo et al., 2025), and HippoRAG and HippoRAG 2 construct graphs over extracted entities and relations and use them during retrieval (Gutiérrez et al., 2024; 2025). Comparative work suggests that graph structure is not uniformly beneficial across retrieval settings (Xiang et al., 2026).

These methods introduce cross-passage structure through coreference normalization, clustering, or graph construction. Some also incorporate local or document-level context. Which entity a mention refers to, and what a passage is about in light of the pages before it, are properties of the document, not of any query, so SAADI performs the conditioning once, at index time. Every extraction step receives two state components accumulated from earlier chunks. One is a selectively recalled active-entity memory. The other is a hierarchical history maintained by a fixed folding schedule. The two components jointly inform extraction from the current chunk.

SLIMRAG is the closest graph-free entity-oriented construction to SAADI (Zhang et al., 2025a). It extracts entities independently from each chunk and aggregates them into a global entity-to-chunk index. At query time, it uses a model to decompose the query and extract entities from the resulting components. These entities are matched against the index and combined with query-chunk similarity. SAADI differs in both stages. Its extraction is conditioned on information selectively recalled from the growing memory and hierarchical history. Its query-time procedure performs no query decomposition or modification. Appendix C.6 isolates the indexing difference by comparing state-conditioned construction with independent chunk-wise construction.

Other work combines generated representations with passage scores while preserving access to source evidence (Xu et al., 2025; Zhou et al., 2026). Weighted multi-field scoring is well established (Robertson et al., 2004), as are score normalization and fusion across channels (Fox & Shaw, 1994; Cormack et al., 2009). Entity information need not improve retrieval by itself. Entity channels can have limited coverage under open-world evaluation (Chatterjee, 2026), and lossy extracted

memories can underperform verbatim evidence (An, 2025). SAADI therefore keeps memory and summaries auxiliary to the passage score and returns only the original passages as evidence.

A.4 Sequential and Evolving Memory

Sequential memory construction has been explored in several settings. SAADI differs in combining additive entity memory with a hierarchically compacted document history and using both to rank unchanged source passages.

MemoRAG forms a global memory representation and uses it after observing the query to generate draft answers or retrieval clues (Qian et al., 2025). Its memory primarily supports model generation rather than directly ranking the original passages of one document. Zep provides a closer example of retrieval-conditioned memory construction (Rasmussen et al., 2025). During ingestion, it retrieves existing entity candidates before resolving new information into a temporal knowledge graph. Its target is persistent agent memory with explicit relations and temporal validity. A-MEM maintains an evolving memory network in which new entries may link to and revise previous representations (Xu et al., 2026), and MemGPT manages a tiered memory through model-issued operations (Packer et al., 2023). Graph-enhanced memories have also been studied for long-context language modelling rather than passage ranking (Alselwi et al., 2026).

SAADI operates on a fixed document rather than an open-ended stream of episodes. Its entity memory grows additively and removes nothing. Its hierarchical history is compacted according to a fixed folding schedule. Neither update is selected by a query-time policy. The state is not returned as evidence and is used only to rank passages that remain unchanged. The fixed schedule also makes the number of index-time model calls determinable from the chunk count and the folding factor f, as derived in Appendix B.5.1. The question studied here is whether this accumulated document state improves retrieval over the passages from which it was built.

A.5 Query-Time and Agentic Retrieval

A separate family modifies retrieval after observing the query. HyDE generates a hypothetical document from the query and retrieves real documents using its embedding (Gao et al., 2023). Iterative approaches interleave retrieval with reasoning or generation. Examples include IRCoT, Self-Ask, FLARE, Iter-RetGen, Self-RAG, and Search-R1 (Trivedi et al., 2023; Press et al., 2023; Jiang et al., 2023; Shao et al., 2023; Asai et al., 2024; Jin et al., 2025). Appendix A.7 describes how these query-side operations can use SAADI as the underlying retrieval interface.

Agentic long-document systems let a model navigate or repeatedly consult structured document representations. Existing methods use gist memories, summary trees, action sequences, graph views, cooperating agents, or learned memory policies (Lee et al., 2024; Chen et al., 2023; Sun et al., 2024; Li et al., 2024b; Zhang et al., 2024b; Yu et al., 2025; Li et al., 2026b; Zhang et al., 2024a). The systems we compare against in Section 5.2 span this range. RAG-R1 and Agentic-R learn when to retrieve again and what to read next (Tan et al., 2025; Liu et al., 2026), A-RAG separates search-result previews from explicit full-chunk reads within its trajectory (Du et al., 2026), and GRIP navigates a structured view of the document (Li et al., 2026a). In all of them, evidence retrieved at one step informs what is sought at the next. This conditioning is reconstructed for every query, so its model cost recurs and the amount of exposed evidence varies across queries, which is why Section 5.2 compares at matched evidence exposure rather than at matched depth or matched call count.

UMA is the closest of these methods in its use of evolving memory (Zhang et al., 2026). It trains an agent end to end and maintains a condensed summary and a structured memory bank whose entries its policy can create, update, delete, and reorganize while producing the answer. SAADI instead grows its entity memory additively, compacts its history on a fixed schedule at index time, uses an off-the-shelf model without task-specific training, and stops at passage retrieval. UMA also pays an offline memory-construction cost that the other agents do not, so Appendix F.4 reports its offline and online resource use separately.

SAADI performs no query expansion, query rewriting, iterative retrieval, or agentic navigation. Once the query arrives, it applies a fixed single-pass ranking over the original chunks. This keeps the query-time mechanism unchanged while the document-side representation is varied.

A.6 Measuring Distributed Evidence

Hit rate at a fixed depth is the standard measure for the retrieval stage, but what it reports depends on how a query's evidence falls across the index. Because retrieval here is a single pass, the returned set either contains the complete evidence for a query or it does not, and a rank-weighted measure records the incomplete case as partial credit.

All-or-nothing set criteria have been used for this reason in multi-hop retrieval. Baleen reports the fraction of claims for which every relevant passage is retrieved (Khattab et al., 2021), and conjunctive retrieval over multi-page documents asks whether all conditions of a query are covered at once (Cha et al., 2026). Chunking benchmarks separately distinguish questions whose evidence is contained in one semantic chunk from those whose evidence spans several (Lu et al., 2026), and QASPER already records which of its questions need more than one paragraph (Dasigi et al., 2021).

Our measure applies that criterion inside a single document over a fixed segmentation. Projecting the annotated spans of a query onto the chunk sequence yields the set of chunks its evidence occupies, which is a single chunk for some queries and a group for others, and the query counts as satisfied only when the returned set contains all of them. One criterion therefore covers both cases, and the difficulty of a query is carried by the size of its target rather than by which population it belongs to. Appendix E.3 gives the definitions and a worked example.

A.7 Composability with Query-Side Methods

SAADI exposes a standard ranking over original passages and returns a top-k set under a budget fixed in advance. It carries no state across queries, makes no query-time LLM call, and does not constrain how the query was produced.

A query-side method can replace or augment Q before Equation (2) is evaluated. All three scoring channels receive Q through the selected retriever, and none requires the query to have been written directly by a user. An iterative controller can call the same retrieval interface several times, with each call returning original passage text. Its additional query-time computation remains external to SAADI and must be accounted for separately. A reranker (Nogueira & Cho, 2019; Nogueira et al., 2020) or reader can likewise operate on the returned passages without depending on how their ranking was produced.

These are properties of the interface rather than empirical results. We do not evaluate these compositions and do not claim that their gains would combine. The narrower claim is that SAADI preserves standard query and passage interfaces while moving document-state construction to index time.

A.8 Robustness under Length, Extension, and Hops

We use *robustness* in a narrow operational sense throughout the paper. It denotes the stability of retrieval quality as long-document conditions become harder while the query-time mechanism remains fixed.

Prior work studies length sensitivity by retaining relevant evidence while increasing surrounding context or by varying the context available to retrieval (Gao et al., 2026; Leng et al., 2024). Benchmarks built on QASPER and LooGLE place one query set at several length levels in the same way (Yang et al., 2025b; Yuan et al., 2024). Such evaluations require careful design. Lexical overlap between questions and evidence can make long-context retrieval artificially easy (Modarressi et al., 2025). Length-manipulation protocols may also change the available signal together with the amount of surrounding content (Arjmandi, 2026). Our extension protocol holds the evidence and query population fixed while adding surrounding material, which isolates the effect of length.

Reasoning complexity provides another axis. Multi-step questions have been shown to degrade faster than single-span questions as context grows, although this result has primarily been measured through answer accuracy rather than retrieval metrics (Dhara & Sheth, 2026). We examine the effect at the retrieval stage through increasing document length, controlled document extension, and increasing hop requirements while leaving query-time retrieval unchanged.

Other robustness dimensions fall outside our experiments. Prior work studies positional sensitivity (Liu et al., 2024; Zeng et al., 2025; Michail et al., 2026; Yu et al., 2026), distractor sensitivity

(Cuconasu et al., 2024; Gabín et al., 2026), and broader forms of retriever robustness (Liu et al., 2025b). We make no claims about these settings.

B METHOD DETAILS

This appendix develops the details omitted from Section 3. We first fix notation (Appendix B.1), then specify the prompt and output format of the state-update call (Appendix B.2) and the exact folding schedule (Appendix B.3), before combining them in the full index-time algorithm (Appendix B.4). Appendix B.5 then derives the number of index-time model calls a document requires, the additional cost of appending one chunk to a document already indexed, and the break-even against methods that incur query-time computation. Configuration choices are ablated separately in Appendix C, and the constructed state is evaluated in Appendix D.

B.1 NOTATION

Table 4 collects the symbols used in Section 3. Index t denotes the current construction step, while i indexes candidate chunks at query time. In the folding derivations below, i is reused locally as a hierarchy level index. Figure 4 shows the index-time pass as a flowchart.

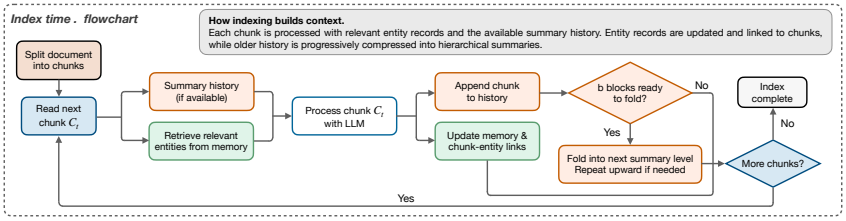

Figure 4: Index-time control flow for a single document. Each chunk in turn is read together with the records recalled from memory and the current summary history, processed by one model call, and used to update both. A fold fires whenever f consecutive units at a level are complete, and it repeats upward while the levels above also complete. The original chunks are never modified, and Algorithm 1 states the same pass as pseudocode.

B.2 PROMPTS AND OUTPUT FORMATS

The index-time pass issues two kinds of model call. The state update of Equation (1) returns the active entities a chunk is about, and the fold of Appendix B.3, Equation (5), compresses f consecutive units into one summary. Each is specified by a system prompt that fixes the task and the output contract, and a user prompt template that serializes the step's inputs. All four are reproduced below.

State update. The system prompt (Figure 5) gives the three inputs distinct roles. The hierarchical history is presented coarse to fine, ending as close as possible to the immediately preceding chunk; the recalled records supply names, aliases and descriptions already held in memory; and the current chunk is the only text from which entities are extracted. The first two may be used solely to resolve names, aliases and coreference, and the prompt forbids inventing facts, attributes, relationships or events. Appendix D.1 measures how well that restriction holds in the constructed state. The user prompt (Figure 6) serializes the three inputs in that order, followed by the output schema. The prompt refers to the raw chunks as L_0, the finest level of the hierarchical context.

Naming and the output contract. An entity already present in the recalled records keeps its stored name exactly, and a new entity receives a short, stable designation that does not collide with a recalled record. Names therefore function as keys. Acronyms and expansions are recorded as aliases of a single entity, not as separate entities, and descriptions state attributes, roles or relations, not alternative names. Each call returns one entry per entity active in the chunk, carrying any aliases and descriptions *newly* supported by that chunk; an entity that is active but adds nothing new is still returned, with empty arrays. A_t is therefore a complete list of the entities the chunk is about, and so

Table 4: Notation used in the SAADI method and method appendix, covering index-time state construction, query-time retrieval, and construction and cost analysis.

Symbol	Meaning	Introduced in
Index-time state construction		
D	Chunk sequence of a document	Setup and state
C_t, C_i	A chunk; the canonical evidence unit	Setup and state
N	Number of chunks in the document	Setup and state
M_t	Active-entity memory after t chunks	Setup and state
H_t	Hierarchical history after t chunks	Setup and state
(M_t, H_t)	The document state	Setup and state
Retrieve	Lexical selector over the memory	Selective recall
R_t	Records recalled from M_{t-1} for C_t	Selective recall
LLM	One state-update model call	State update
A_t	Active entities returned for C_t; their names are stored as the chunk's links into the memory	State update
$\oplus$	Merge: opens a record for an unseen entity, otherwise extends the existing one	State update
f	Folding factor of the summary hierarchy	Hierarchical folding
$L_i^{(j)}$	The j-th level-i summary	Hierarchical folding
Summarize	The fold operation	Hierarchical folding
Query-time retrieval		
Q	The query	Three scoring channels
S_{text}	Text-channel score of Q against a chunk	Three scoring channels
S_{mem}	Memory-channel score	Three scoring channels
S_{sum}	Summary-channel score	Three scoring channels
$\text{Score}_T, \text{Score}_M, \text{Score}_S$	Names used in the appendix tables for S_{text}, S_{mem} and S_{sum}	Normalization and fusion
$\text{Score}_N, \text{Score}_A, \text{Score}_D$	Names used in the appendix tables for the three field terms of Equation (11), each already maximized over E_i	Memory channel
E_i	Records the memory holds for the names linked to C_i	Memory channel
s_N, s_A, s_D	Score of Q against a record's name, aliases, descriptions	Memory channel
μ_N, μ_A, μ_D	Field weights inside the memory channel	Memory channel
$\hat{S}$	A channel score after z-score standardization	Normalization and fusion
$\lambda_T, \lambda_M, \lambda_S$	Channel weights in the fused score	Normalization and fusion
S_{SAADI}	Fused score used for ranking	Normalization and fusion
k	Number of chunks returned	Retrieval output
Construction cost		
$T(N, f)$	Index-time model calls for a document	Index-time calls
$\bar{c}$	Mean model calls per query of an alternative method	Break-even
$Q^\star$	Break-even number of queries	Break-even

a complete set of links, while $\oplus$ stays purely additive. Only new information crosses into M_t, and the call never rewrites what memory already holds.

Span anchors. Alongside the entity records, each call returns a span anchor for every active entity; the exact mention in the chunk where that entity was found, together with its immediate left and right context. The prompt phrases an anchor as the position at which a canonical name would be placed, but no such insertion is performed. The text that is indexed and returned is byte-identical to the original chunk, and the anchors are never seen at query time. We record them because they identify which part of a chunk each entity was extracted from. That matters when one entity is active at several points in a chunk, and a later study of indexing and retrieval at different granularities will need it.

Hierarchical summarization. The fold prompt (Figure 7) makes no reference to a level. It condenses one group of f units into a single concise, factual sentence of at most sixty words, keeping core entities and events and adding no speculation. The user prompt (Figure 8) carries the items

being folded together with the folding factor and the position of the fold in the schedule of Appendix B.3. The same pair therefore serves every level, so a level-two fold summarizes level-one summaries with the prompt that summarizes chunks.

```
You extract active entities from one current text chunk.

Goal:
- Return `active_entities` and one placement `insertion` for every active entity in CURRENT_CHUNK.

Inputs:
- HIERARCHY_CONTEXT_JSON is the already processed history before CURRENT_CHUNK. It is arranged from coarser summaries to finer L0 chunks, ending as close as possible to the immediate previous chunk.
- PREVIOUS_ACTIVE_ENTITIES_JSON contains known entity names, aliases, and descriptions from memory that might help for clarification of the current chunk.
- CURRENT_CHUNK is the only text you extract active entities from.

Output:
- Return JSON only with keys `active_entities` and `insertions`.
- Do not output CURRENT_CHUNK, rewritten chunk text, summaries, or any extra keys.

Grounding rules:
- Use HIERARCHY_CONTEXT_JSON and PREVIOUS_ACTIVE_ENTITIES_JSON only to resolve names, aliases, and coreference.
- Insertion anchors must be grounded in CURRENT_CHUNK.
- Do not invent facts, attributes, relationships, or events.

Span-copy rules (CRITICAL):
- `anchor_text`, `left_context`, and `right_context` MUST be exact substrings copied from CURRENT_CHUNK.
- Do NOT modify spelling, punctuation, or whitespace when copying spans.
- Use the minimal left/right context necessary to uniquely identify the anchor.

Naming rules:
- If an entity already exists in PREVIOUS_ACTIVE_ENTITIES_JSON, reuse its exact stored name; never rename it or mint a variant, even when a new description appears. Only treat a current mention as that entity when it genuinely refers to the same entity. The rules below govern only how to name a NEW entity.
- Name a new entity by its own standard designation, the term that denotes the concept itself, never by a citation or reference marker that merely points to it (e.g., "the model in [12]" is named by the model, never "[12]").
- Give each new entity a unique, stable name that does not collide with a different entity in PREVIOUS_ACTIVE_ENTITIES_JSON. The name need not be an exact substring of CURRENT_CHUNK; it must be a short, clear designation. By default use its standard designation (for named methods, datasets, or models often the acronym, e.g. "BERT", "BM25"; for general concepts the full term, e.g. "large language model").
- If the default name would collide with a different entity already in memory (e.g. two entities share the acronym "ML"), qualify it into a unique short form (e.g. "Meta Learning (ML)") and keep the bare form as an alias.
- Represent each entity once. Acronyms, expansions, and other surface forms of the same entity are aliases of that one entity, never separate entities.
- Always record both the standard full name and the acronym/short form in `aliases`, even when one of them is identical to the name.

Insertion rules:
- `insertions` must contain exactly one entry for each item in `active_entities`, with `name` equal to that active entity's `name`.
- `anchor_text` should be the mention, alias, pronoun, or referring phrase in CURRENT_CHUNK where that entity participates.
- The system places ` (name)` immediately after `anchor_text`: only the canonical name is inserted, and you do not output the inserted text.

`active_entities` rules:
- Return every entity that is active, present, or participating in CURRENT_CHUNK.
- For new or updated entities, include only aliases/descriptions newly supported by CURRENT_CHUNK.
- For entities that are only present with no new aliases/descriptions, include the name with empty aliases and description arrays.
- If no entities are active in CURRENT_CHUNK, return [].
- Descriptions state attributes, roles, or relations of the entity -- never an alternative name for it. Keep them short and factual. A citation or reference marker may be appended to a description that states real content, but must never be a description on its own.

Output contract:
- `insertions` must be [] only when `active_entities` is [].
- Each insertion must contain:
  - `name`: exact active entity name from `active_entities`
  - `anchor_text`: exact mention from CURRENT_CHUNK that receives the parenthetical
  - `left_context`: exact text immediately before `anchor_text` copied from CURRENT_CHUNK
  - `right_context`: exact text immediately after `anchor_text` copied from CURRENT_CHUNK
```

Figure 5: System prompt for the state-update call. It defines the active-entity extraction task, grounding and naming constraints, and the record and span-anchor output contract.

```
[HIERARCHY_CONTEXT_JSON]
{hierarchy_context}

[PREVIOUS_ACTIVE_ENTITIES_JSON]
{filtered_entities}

[CURRENT_CHUNK]
{chunk_text}

[OUTPUT_SCHEMA]
{"active_entities":[{"name":"string","description":["string"],"aliases":["string"]}],"insertions":[{"name":"same string as the active entity
name","anchor_text":"string","left_context":"string","right_context":"string"}]}

[INSTRUCTIONS]
Return JSON only, with keys active_entities and insertions and no other text. Include exactly one insertion for each active entity, and do not include any inserted-text field. Then [END_JSON].
```

Figure 6: User prompt for the state update. It serializes the recalled records R_t, the hierarchical history H_{t-1}, and the current chunk C_t.

```
You summarize one branching-factor group into one concise, factual summary.
Keep core entities and events. Do not add speculation.
Output plain text only, no bullets, no markdown, no JSON.
Terminate with [END_SUMMARY].
```

Figure 7: System prompt for hierarchical folding. It defines the summarization task over one branching-factor group and the plain-text output contract.

B.3 Folding Schedule

Folding is driven by the step counter alone. Once C_t has been appended to the history, a fold fires at level i exactly when f^i divides t. It replaces the f consecutive level-$(i-1)$ units that have just

```
[ROLLUP_META]
{"branching_factor": <branching_factor>, "lower_cycle": <lower_cycle>, "next_cycle": <next_cycle>, "step": <step>}

[ROLLUP_ITEMS_JSON]
<items_as_json_array>

[INSTRUCTIONS]
Summarize the items into one concise sentence (<= 60 words). Output plain text only, then [END_SUMMARY].
```

Figure 8: User prompt for hierarchical folding. It provides the f consecutive units being folded together with the metadata identifying their position in the folding schedule.

become available with one level-i summary,

$$L_i^{(j)} \;=\; \mathrm{Summarize}\big(L_{i-1}^{(f(j-1)+1)}, \ldots, L_{i-1}^{(fj)}\big), \tag{5}$$

where $L_i^{(j)}$ is the j-th summary at level i and level zero denotes the raw chunks, $L_0^{(j)} = C_j$. Several levels can fire in the same step, because a level-i fold may complete a group of f level-i units, which is then folded at level $i{+}1$.

Worked example. Take $f = 5$. Nothing folds through the first four steps. At $t = 5$ the raw chunks $C_1, \ldots, C_5$ are replaced by $L_1^{(1)}$, and the same happens at $t = 10$, 15 and 20, producing $L_1^{(2)}$ through $L_1^{(4)}$. Step $t = 25$ is the first in which two levels fire. Appending C_{25} completes $C_{21}, \ldots, C_{25}$, so those five chunks are folded into $L_1^{(5)}$. Five level-one summaries are now available, so $L_1^{(1)}, \ldots, L_1^{(5)}$ are folded in turn into $L_2^{(1)}$. The step therefore costs two summarization calls rather than one. Its foldable hierarchy is reduced to a single level-two summary covering $C_1, \ldots, C_{25}$, while C_{25} is retained verbatim for the next state update.

What the history holds. The foldable part of the history after step t can be read off the base-f representation of t. Writing $t = \sum_{i \geq 0} d_i f^i$, it holds exactly d_i units at level i. With $f = 5$ and $t = 40$, the foldable hierarchy holds one level-two summary covering $C_1, \ldots, C_{25}$, three level-one summaries covering $C_{26}, \ldots, C_{40}$, and no unfolded chunks. The number of foldable units is therefore the digit sum of t.

The retained chunk. One additional unit sits outside this accounting. The chunk immediately preceding the current step is always carried verbatim into the next state update, even when a fold has just absorbed it. At $t = 25$ with $f = 5$ both a level-one and a level-two fold fire, and C_{25} is still passed into step 26 alongside the summaries, so the incoming chunk is read against the text that directly precedes it rather than against a compressed account of it.

Size of the history. Folds happen every f steps, then every f^2, then every f^3, so the levels of the hierarchy are indexed by powers of f. After t chunks only the levels with enough material beneath them can exist. A level-x summary needs f^x chunks, so the highest level present at step t is the x satisfying

$$f^x \;\leq\; t \;<\; f^{x+1}.$$

Taking logarithms in base f gives $x \leq \log_f t < x + 1$, and therefore $x = \lfloor \log_f t \rfloor$.

Now count what the history can hold at each of those levels. A level never holds f units, because the f-th one triggers a fold and is replaced by a single unit one level up, so every level holds at most $f - 1$. There are x summary levels together with the raw chunks. Writing H_t^{fold} for the foldable part of the history,

$$|H_t^{\mathrm{fold}}| \;\leq\; (f-1)\big(\lfloor \log_f t \rfloor + 1\big).$$

The history therefore grows with the logarithm of the prefix rather than with the prefix itself. With $f = 5$, a document of ten thousand chunks reaches level five, so the foldable hierarchy holds at most twenty-four units; the state-update context may contain one additional retained predecessor chunk. The same x fixes how many terms appear in the call count of Appendix B.5.1.

Every foldable unit is either a bounded raw chunk or a summary that the fold prompt limits to sixty words, and at most one additional predecessor chunk is retained. The total history context therefore

Algorithm 1 Index-time state construction for one document.

Require: chunk sequence $D = (C_1, \ldots, C_N)$, folding factor f
Ensure: memory M_N, hierarchical history H_N, per-chunk name links, level-one summaries $\mathcal{L}_1$

$M_0 \leftarrow \emptyset, \quad H_0 \leftarrow \emptyset, \quad \mathcal{L}_1 \leftarrow ()$
for $t = 1$ **to** N **do**
  $R_t \leftarrow \mathrm{Retrieve}\big(C_t,\, M_{t-1}\big)$ ▷ selective recall, Section 3.1
  $A_t \leftarrow \mathrm{LLM}\big(C_t,\, R_t,\, H_{t-1}\big)$ ▷ Eq. (1), one model call
  $M_t \leftarrow M_{t-1} \oplus A_t$
  store $\mathrm{names}(A_t)$ as the links of C_t
  $F_t \leftarrow$ the foldable hierarchy in H_{t-1} with C_t appended ▷ discard any retained predecessor copy
  $i \leftarrow 1$
  while $f^{\,i}$ divides t **do**
    $G \leftarrow$ the f consecutive level-$(i-1)$ units just completed in F_t
    $u \leftarrow \mathrm{Summarize}(G)$ ▷ Eq. (5), one model call
    replace G in F_t by u
    if $i = 1$ **then**
      append u to $\mathcal{L}_1$ ▷ kept for the summary channel after later folds
    end if
    $i \leftarrow i + 1$
  end while
  $H_t \leftarrow F_t$
  if C_t is no longer present verbatim in F_t **then**
    retain a non-foldable verbatim copy of C_t in H_t
  end if
end for
return M_N, H_N, the links, and $\mathcal{L}_1$

remains $O(\log t)$. Figure 9 in Appendix D.2.2 tracks the summary portion of this context over the course of a document.

The folding factor therefore fixes both how fast the history is compacted and how deep it becomes. Appendix C.4 reports how retrieval responds when it is varied.

B.4 Index-Time Algorithm

Algorithm 1 collects in one place what Figure 4 shows as a flowchart and Section 3.1 describes step by step, fixing the order in which the operations of a step occur. Only two lines issue a model call, the state update and the fold, and Appendix B.5.1 counts them. The level-one summaries are collected in $\mathcal{L}_1$ as they are produced, since the history replaces them once they are folded into a higher level, and the summary channel of Section 3.2 scores against them.

B.5 Cost Analysis

B.5.1 Index-Time Call Count

A document of N chunks is read as a sequence of N steps, and by the argument of Appendix B.3 its hierarchy reaches level $x = \lfloor \log_f N \rfloor$. Algorithm 1 issues a model call in exactly two places, so it is enough to count each separately and add.

State updates. Every pass through the outer loop performs one state update, with no condition attached to it, so the N chunks account for exactly N calls.

Folds. The levels of the hierarchy fire on their own schedules. Level one fires once every f steps, level two once every f^2 steps, and so on up to level x. Over N steps the number of times each level

fires is the number of its multiples that do not exceed N,

level	1	2	$\cdots$	x
fires every	f steps	f^2 steps	$\cdots$	f^x steps
calls	$\lfloor N/f \rfloor$	$\lfloor N/f^2 \rfloor$	$\cdots$	$\lfloor N/f^x \rfloor$

so building the whole hierarchy costs $\sum_{i=1}^{x} \lfloor N/f^i \rfloor$ calls.

Total. Adding the two contributions gives

$$T(N, f) \;=\; N \;+\; \sum_{i=1}^{\lfloor \log_f N \rfloor} \Big\lfloor \frac{N}{f^i} \Big\rfloor, \tag{6}$$

which restates Equation (3) of Section 3.3. Every quantity on the right is fixed once N and f are known, so the count is determined before the pass begins and does not depend on the content of the document or on the queries later issued against it.

Upper bound. Dropping the floors raises every term, and letting the sum run to infinity raises it again, which leaves a geometric series with ratio $1/f$,

$$\sum_{i=1}^{\lfloor \log_f N \rfloor} \Big\lfloor \frac{N}{f^i} \Big\rfloor \;<\; \sum_{i=1}^{\infty} \frac{N}{f^i} \;=\; N \cdot \frac{1/f}{1 - 1/f} \;=\; \frac{N}{f-1},$$

and therefore

$$T(N, f) \;<\; N + \frac{N}{f-1} \;=\; \frac{f}{f-1}\, N, \tag{7}$$

which at $f = 5$ is $1.25N$. The whole summary hierarchy therefore contributes fewer than a quarter of a call per chunk on top of the state update.

Tightness. Only two things separate $T(N, f)$ from that bound. Each floor rounds away less than one call, and the terms beyond level x, added only to complete the geometric series, are smaller still. Together they amount to $O(\log_f N)$, so $T(N, f) = \frac{f}{f-1} N - O(\log_f N)$. For example, with $N = 94$ and $f = 5$, Equation (6) gives $T = 94 + \lfloor 94/5 \rfloor + \lfloor 94/25 \rfloor = 115$, compared with the upper bound 117.5.

Worked example. Take $f = 5$ and $N = 746$. The hierarchy reaches $x = \lfloor \log_5 746 \rfloor = 4$, since $625 \le 746 < 3125$, and the levels fire $\lfloor 746/5 \rfloor = 149$, $\lfloor 746/25 \rfloor = 29$, $\lfloor 746/125 \rfloor = 5$ and $\lfloor 746/625 \rfloor = 1$ times, for 184 summaries. The document costs $746 + 184 = 930$ calls, against a bound of 932.5.

Verification. Equation (6) predicts both the number of summaries a document receives and the total number of index-time calls it costs, and both are measured directly. Table 5 compares prediction against measurement. The small residuals come from averaging, since the mean of $\lfloor N_d/f^i \rfloor$ over documents differs from the same expression evaluated at the mean document length.

B.5.2 Cost of Appending a Chunk

Nothing in Algorithm 1 revisits a chunk once it has been processed. A new chunk is scored against the existing memory, folded into the existing history, and its entities are merged additively, so a document that grows is extended, not rebuilt. This subsection counts what one more chunk costs.

The count. Suppose the document holds N chunks and C_{N+1} arrives. The extraction call always fires, so the step costs at least one. If $N + 1$ is not a multiple of f then no group is complete at any level and that single call is the whole cost. Otherwise at least one fold fires. The levels reachable at this step go up to $\lfloor \log_f (N+1) \rfloor$, one higher than before whenever $N + 1$ is the first step to reach a new power of f, and a fold at level i fires exactly when f^i divides $N + 1$. The step therefore costs

$$1 \;+\; \sum_{i=1}^{\lfloor \log_f (N+1) \rfloor} \mathbf{1}\big[\, f^i \mid N+1 \,\big] \tag{8}$$

calls.

Table 5: Analytical and measured index-time call cost per document at $f = 5$. The predicted columns evaluate Equation (6) at the mean chunk count of each corpus, and the measured columns report the cached construction runs. Total is the chunk count plus the summary count in both cases, and the two summary counts differ because a mean of floors is not the floor of a mean.

Dataset	Chunks / doc	Summaries / doc.		Total calls / doc.	
		Predicted	Measured	Predicted	Measured
LooGLE	31.86	7	6.67	38.86	38.53
MuSiQue-32K	380.04	94	93.16	474.04	473.20
QASPER-64K	746.04	184	184.57	930.04	930.61
NovelHopQA	429.78	105	105.44	534.78	535.22

Why the logarithm is not the answer. The logarithm fixes how many terms the sum has, not how many of them are on. With $f = 5$ the steps $N + 1 = 25$ and $N + 1 = 35$ both satisfy $\lfloor \log_5 25 \rfloor = \lfloor \log_5 35 \rfloor = 2$, so both sums run over $i = 1$ and $i = 2$, yet they do not cost the same,

$N+1$	$\mathbf{1}\big[5 \mid N+1\big]$	$\mathbf{1}\big[25 \mid N+1\big]$	folds
25	1	1	2
35	1	0	1

At 25 the level-one fold produces $L_1^{(5)}$, the fifth level-one summary, so a level-two fold fires as well and the step costs three calls. At 35 only two level-one summaries have accumulated since $L_2^{(1)}$ was formed, so the step costs two.

Evaluating the sum. The indicators cannot switch on and off freely. If f^2 divides $N + 1$ then f divides it as well, and the same holds at every level, so the indicators form a run of ones followed by zeros and the sum is the length of that run. That length is the number of times f divides $N + 1$, the number of trailing zeros of $N + 1$ written in base f: two for 25, one for 35, none for 27.

Worst case and average. A run cannot be longer than the number of levels, so the sum never exceeds $\lfloor \log_f(N + 1) \rfloor$. This bound is attained whenever $f^{\lfloor \log_f(N+1) \rfloor}$ divides $N + 1$, in particular when $N+1$ is a power of f. Averaged over a long run of appended chunks, level i fires on a fraction f^{-i} of steps. Summing over levels gives the long-run average number of folds,

$$\frac{1}{f} + \frac{1}{f^2} + \frac{1}{f^3} + \cdots \;=\; \frac{1/f}{1 - 1/f} \;=\; \frac{1}{f-1},$$

and adding the extraction call that always fires gives an amortized limit of $f/(f - 1)$ calls per appended chunk. At $f = 5$ this approaches 1.25, consistent with Equation (6) divided by N as N grows.

Appending is therefore bounded. A single chunk costs one call in four steps out of five at $f = 5$, never more than $1 + \lfloor \log_f(N + 1) \rfloor$, and $f/(f - 1)$ calls amortized, independent of how much of the document has already been processed.

B.5.3 Break-Even Against Per-Query Computation

SAADI spends LLM calls while the document is indexed and none when a query arrives; a method that invokes an LLM at query time pays per query instead. Which is cheaper depends on how many queries are issued against the document, so the question is where their totals cross.

The two totals. Suppose the same document is queried Q times. SAADI has already paid $T(N, f)$ calls and pays nothing further, so its total is

$$\mathrm{SAADI}(Q) \;=\; T(N, f),$$

a flat line that does not move with Q. An alternative that averages $\bar{c}$ calls on each query has spent

$$\mathrm{Alternative}(Q) \;=\; \bar{c}\,Q,$$

a line through the origin with slope $\bar{c}$.

Where they cross. Setting the two equal and solving for Q gives

$$\bar{c}\,Q^\star \;=\; T(N,f) \qquad\Longrightarrow\qquad Q^\star \;=\; \frac{T(N,f)}{\bar{c}} \;\approx\; \frac{f}{f-1}\cdot\frac{N}{\bar{c}}, \tag{9}$$

where the approximation uses the asymptotic form of $T(N,f)$ derived above. Below $Q^\star$ the alternative has spent fewer calls on this document, and above it SAADI has.

Reading the formula. Two things follow from the shape of Equation (9). The break-even grows linearly with document length, so a long document needs more queries before its index repays itself. It also falls as $\bar{c}$ grows, so the more computation an alternative spends on each query, the sooner SAADI overtakes it.

Worked example. A LooGLE document costs $T = 38.53$ calls to index. An alternative averaging $\bar{c} = 2.43$ calls per query therefore matches that total after $38.53/2.43 \approx 16$ queries against the same document. For QASPER-64K, indexing instead costs $T = 930.61$ calls, and against an alternative averaging $\bar{c} = 3.34$ calls per query the break-even moves out to about 279 queries. Section 5 reports these values for every method and dataset.

Alternatives that also build an index. Some methods pay an offline cost of their own. Writing U^{off} for that cost and U^{on} for their mean per-query use, equality occurs at a nonnegative query count when $U^{\text{off}} < T(N,f)$. Defining the crossover threshold over nonnegative query counts gives

$$Q^\star \;=\; \max\Big\{\,\frac{T(N,f)-U^{\text{off}}}{U^{\text{on}}},\;0\,\Big\}. \tag{10}$$

The maximum covers the case where the alternative's own index already costs more than SAADI's, so that it is never cheaper and the break-even sits at zero queries.

Other resources. Nothing in the argument is specific to counting calls. Replacing T and $\bar{c}$ by the corresponding totals in input tokens, or in generated tokens, gives the break-even in those units instead. These are resource-equivalent crossovers rather than monetary ones, since a call, an input token and a generated token are priced differently and the prices depend on the model. For model calls, $T(N,f)$ follows analytically from Algorithm 1, while $\bar{c}$ is measured. Token-based crossovers instead use measured offline and per-query token totals for the corresponding methods.

C Construction Ablation Study

Section 3 fixes a number of settings without arguing for them. This appendix varies each in turn, holding everything else at the default of Section 4, and reports hit rate on the datasets for which the configuration was run. Unless stated otherwise, values are percentages, best and second-best distinct values are marked, and † marks the default configuration.

C.1 Active-Entity Budget

Selective recall exposes a bounded number of records, each carrying a bounded number of aliases and descriptions. The default budget, written $E/A/D$, is five records, two aliases and three descriptions. Table 6(a) varies it at HR@5 on LooGLE and NovelHopQA under BM25 and Contriever. The four budgets lie within 2.7 points of each other in every block, and the default is first or second in three of the four blocks and third in the fourth, so it is not a sharply tuned setting.

C.2 Active-Entity Selection Method

The recall step scores the incoming chunk against the recorded entity text, and Table 6(b) compares two ways of doing it, plain word overlap and BM25. The two are close, and BM25 is ahead in all four blocks, by 0.2 to 1.2 points.

Table 6: Active-entity configuration ablations at HR@5, with † marking the default in each panel. Panel A varies the recall budget $E/A/D$, the maximum numbers of active entities, aliases per entity and descriptions per entity, and Panel B varies the selector that scores an incoming chunk against the recorded entity text. Best and second-best values are marked within each row of Panel A, and the best value within each row of Panel B.

(A) Active-entity budget

Dataset	Retriever	HR@5			
		3/2/5	5/2/3†	5/2/5	7/2/5
LooGLE	BM25	**93.0**	92.3	90.3	**92.8**
	Contriever	74.4	**75.4**	**74.9**	74.1
NovelHopQA	BM25	17.7	**18.1**	**18.5**	17.8
	Contriever	**13.7**	**13.5**	12.9	13.1

(B) Active-entity selector

Dataset	Retriever	HR@5	
		Word	BM25†
LooGLE	BM25	91.1	**92.3**
	Contriever	75.2	**75.4**
NovelHopQA	BM25	17.8	**18.1**
	Contriever	13.1	**13.5**

Table 7: Robustness to the model used for index-time construction. Only the construction model changes, with the prompts, the state update and the retrieval pipeline held fixed. Best and second-best values are marked within each dataset and retriever block.

Dataset	Retriever	Construction LLM	HR@5	HR@10
LooGLE	BM25	Qwen2.5-7B	**91.6**	**95.7**
		Mistral-Small-3.1-24B	91.4	95.5
		Qwen3-30B-A3B†	**92.3**	**96.5**
		Llama-3.3-70B	91.0	95.2
	Contriever	Qwen2.5-7B	72.3	**84.8**
		Mistral-Small-3.1-24B	**73.1**	84.3
		Qwen3-30B-A3B†	**75.4**	**85.7**
		Llama-3.3-70B	72.2	84.3
NovelHopQA	BM25	Qwen2.5-7B	**18.0**	**21.5**
		Mistral-Small-3.1-24B	17.0	20.8
		Qwen3-30B-A3B†	**18.1**	**22.0**
		Llama-3.3-70B	17.3	21.3
	Contriever	Qwen2.5-7B	**13.3**	18.0
		Mistral-Small-3.1-24B	11.3	**18.3**
		Qwen3-30B-A3B†	**13.5**	**18.2**
		Llama-3.3-70B	11.8	17.4

C.3 Construction LLM Robustness

The state is built by a model, so the question is how much of the retrieval quality depends on which model builds it. Table 7 repeats the construction with several models and leaves every other setting fixed. Retrieval is stable under that choice, and more so at HR@10 than at HR@5. Across the four dataset and retriever blocks, LooGLE and NovelHopQA under BM25 and Contriever, the three alternatives move HR@10 by at most 1.4 points from the default, and the 7B model stays within 0.9. At HR@5 the spread is wider and reaches 3.2 points on LooGLE under Contriever, where every alternative falls below the default. The construction model therefore changes the result by less than the choice of retriever does.

C.4 Folding Factor

The folding factor sets how quickly the history is compacted and how deep the hierarchy becomes, and through Equation (6) it also sets the construction cost. Table 8 varies it over $f \in \{2, 5, 8, 10\}$. $f = 5$ is best or tied in fourteen of the sixteen columns and within 0.1 of the best in the other two, and the larger factors 8 and 10 are the weakest. Sensitivity to the factor is dataset-dependent. LooGLE and QASPER-64K move little across the range, while MuSiQue-32K spans eight points of HR@5 under BM25, which is where the preference for $f = 5$ is decided.

Table 8: Effect of the hierarchical-summary folding factor, with † marking the default $f = 5$. All other SAADI settings are held fixed. Best and second-best values are marked within each dataset block of a column, and tied values share a mark.

Dataset	f	BM25		Contriever	
		HR@5	HR@10	HR@5	HR@10
LooGLE	2	**92.2**	96.1	**74.2**	**85.5**
	5†	**92.3**	**96.5**	**75.4**	**85.7**
	8	90.8	96.0	72.2	84.7
	10	91.3	**96.6**	73.8	84.9
MuSiQue-32K	2	**57.9**	74.8	**62.3**	78.9
	5†	**62.8**	**77.7**	**68.8**	**83.7**
	8	**57.9**	73.8	61.7	78.8
	10	54.4	**75.1**	61.3	**79.1**
QASPER-64K	2	**16.1**	**26.0**	**15.9**	**25.3**
	5†	**17.0**	**26.1**	**16.2**	**25.8**
	8	15.8	25.1	15.4	23.9
	10	15.6	25.2	**15.9**	24.5
NovelHopQA	2	16.4	**21.6**	**13.3**	17.7
	5†	**18.1**	**22.0**	**13.5**	**18.2**
	8	**17.2**	20.7	**13.5**	**18.3**
	10	16.9	20.7	11.3	16.9

C.5 Fusion Weight Selection

Fusion happens at two levels. The memory channel combines the three record fields with weights (μ_N, μ_A, μ_D), and the final score combines the three channels with weights $(\lambda_T, \lambda_M, \lambda_S)$. The two are selected in order rather than independently. The channel weights are fixed first, with the memory channel scored on entity names alone, and the field weights are then chosen with the channel weights held at that setting. The way the memory score is formed and normalized is varied last.

Because the two weight sweeps below require evaluating many configurations, Tables 9 and 10 are evaluated on a fixed 100-query sample rather than the full evaluation set. The sample is drawn from the LooGLE and NovelHopQA evaluation sets and is not held out from the results of Section 5; the selected weights are then held fixed for every dataset, retriever, depth and condition, so MuSiQue-32K and QASPER-64K take no part in the selection. All configurations within each table use the same sample, so comparisons are paired within the table; their absolute values should not be compared directly with full-set ablations.

C.5.1 Level 1, Channel Weights

The fused score of Equation (2) weights the text, memory and summary channels by $(\lambda_T, \lambda_M, \lambda_S)$. Table 9 varies them around the default $(1.0, 0.5, 0.3)$ on the fixed 100-query sample described above. The default is best or tied in every column, and setting the three weights equal costs 2 to 13 points, which supports the arrangement where the chunk text anchors the ranking and the other two adjust it.

C.5.2 Level 2, Memory-Channel Weights

Inside the memory channel, each record field is scored separately. Writing E_i for the records linked to C_i and $s(Q, \cdot)$ for the base retriever's score,

$$S_{\text{mem}}(Q, C_i) \;=\; \mu_N \max_{e \in E_i} s\big(Q, \text{name}(e)\big) \;+\; \mu_A \max_{e \in E_i} s\big(Q, \text{alias}(e)\big) \;+\; \mu_D \max_{e \in E_i} s\big(Q, \text{desc}(e)\big). \tag{11}$$

Table 10 varies the field weights around the default $(1, 1, 1)$ on the same fixed 100-query sample. On LooGLE the equal weighting is best in all six columns and every other setting is 1 to 9 points below it, so all three fields carry signal there. On NovelHopQA the nine settings lie within one point of each other. Over both datasets the equal weighting is best or tied in ten of the twelve columns and one point below in the other two.

Table 9: Ablation of the channel weights $(\lambda_T, \lambda_M, \lambda_S)$ over the chunk-text, memory and summary channels, with † marking the chosen $(1.0, 0.5, 0.3)$. This is the first of two selection stages: the field weights inside the memory channel are held at their default $(1, 1, 1)$ here, and Table 10 varies them at the chosen channel weights. All configurations use the same fixed sample of 100 queries per dataset, which is the sample the weights were selected on.

Dataset	$(\lambda_T, \lambda_M, \lambda_S)$	HR@5			HR@10		
		BM25	Contriever	Jina	BM25	Contriever	Jina
LooGLE	$(1, .5, .3)^\dagger$	92	75	70	97	86	82
	$(1, 1, 1)$	87	62	64	94	73	78
	$(1, .3, 1)$	89	68	69	95	82	82
	$(1, .5, 1)$	90	67	68	95	79	81
	$(1, 1, .5)$	87	61	63	94	73	79
NovelHopQA	$(1, .5, .3)^\dagger$	18	14	15	22	18	21
	$(1, 1, 1)$	16	11	12	19	16	16
	$(1, .3, 1)$	16	11	13	21	17	19
	$(1, .5, 1)$	16	12	12	21	16	18
	$(1, 1, .5)$	16	11	13	20	15	16

Table 10: Ablation of the field weights (μ_N, μ_A, μ_D) inside the memory channel, which weight matches to entity names, aliases, and descriptions before the memory score enters the fusion. This is the second selection stage, with the channel weights held at the values chosen in Table 9. † marks the default $(1, 1, 1)$, and all configurations use the same fixed sample of 100 queries per dataset.

Dataset	(μ_N, μ_A, μ_D)	HR@5			HR@10		
		BM25	Contriever	Qwen	BM25	Contriever	Qwen
LooGLE	$(1, 0, 0)$	90	69	69	95	79	82
	$(1, .3, 0)$	90	69	69	95	79	81
	$(1, .5, 0)$	90	68	68	95	79	82
	$(1, 0, .3)$	90	70	72	96	81	83
	$(1, 0, .5)$	90	71	72	95	83	84
	$(1, .3, .3)$	89	69	72	96	82	83
	$(1, .5, .3)$	89	69	71	96	82	83
	$(1, 1, 1)^\dagger$	92	75	77	97	86	87
	$(0, 1, 1)$	90	70	74	96	83	85
NovelHopQA	$(1, 0, 0)$	18	13	16	22	17	22
	$(1, .3, 0)$	17	13	16	21	18	22
	$(1, .5, 0)$	17	13	15	22	18	22
	$(1, 0, .3)$	18	13	16	22	17	22
	$(1, 0, .5)$	17	13	16	21	18	22
	$(1, .3, .3)$	17	13	16	22	18	21
	$(1, .5, .3)$	17	13	16	22	18	21
	$(1, 1, 1)^\dagger$	18	14	15	22	18	21
	$(0, 1, 1)$	18	13	16	22	17	22

C.5.3 Score Formulation and Normalization

Two further choices remain in how the memory channel is built. The default formulation, Equation (11), keeps the three fields apart and takes the strongest match within each before combining them. The linearized variant removes the boundary between the fields. Each record is flattened into a single representation $\hat{e}$ holding its name, aliases and descriptions together, and the chunk receives the score of its strongest record,

$$S^{\text{lin}}_{\text{mem}}(Q, C_i) \;=\; \max_{e \in E_i} s\big(Q, \hat{e}\big). \tag{12}$$

The difference is not only notational. Component-wise scoring allows the three maxima to come from three different records, and it keeps μ as a way of stating what a name match is worth against a description match. Linearized scoring gives up both, since a single record supplies the whole score and μ has nothing left to weight.

Table 11: Ablation of the memory-score formulation and its normalization, with † marking the default. Component-wise scoring keeps the name, alias, and description matches separate before combining them, while linearized scoring treats each record as one concatenated representation. Best and second-best values are marked within each formulation block of a column.

Dataset	Memory-score formulation	Norm.	HR@5		HR@10	
			BM25	Contriever	BM25	Contriever
LooGLE	Component-wise fusion	Raw	91.7	72.8	**95.7**	84.2
		Min–max	**91.8**	**73.9**	**95.7**	**85.4**
		Z-score†	**92.3**	**75.4**	**96.5**	**85.7**
	Linearized memory	Raw	91.2	73.3	95.5	85.5
		Min–max	**91.8**	**74.1**	**95.6**	**85.8**
		Z-score	**92.1**	**74.5**	**95.9**	**86.1**
NovelHopQA	Component-wise fusion	Raw	**18.4**	**14.0**	**22.0**	**18.5**
		Min–max	**18.1**	**13.5**	**22.1**	**18.4**
		Z-score†	**18.1**	**13.5**	**22.0**	18.2
	Linearized memory	Raw	**18.4**	10.0	**22.0**	**15.7**
		Min–max	**18.4**	**11.3**	**21.6**	**16.2**
		Z-score	**18.4**	**11.1**	**22.0**	15.3

Each channel is then standardized over the candidate chunks before fusion. Writing S for a channel's raw scores across that population,

$$\hat{S}^{\text{raw}} = S, \qquad \hat{S}^{\text{min-max}} = \frac{S - \min S}{\max S - \min S}, \qquad \hat{S}^{\text{z}} = \frac{S - \text{mean}(S)}{\text{std}(S)}. \tag{13}$$

Table 11 crosses the two formulations against the three normalization schemes. The default, component-wise scoring with the z-score, is ahead of the best linearized variant in five of the eight columns, by up to 2.2 points on NovelHopQA under Contriever, level in one, and behind by at most 0.4 in the other two. Within component-wise scoring, the z-score is best in every LooGLE cell and costs nothing to apply, while on NovelHopQA the three schemes sit within half a point of each other, so it is adopted as the default without uniformly dominating the alternatives.

C.6 State-Conditioning Constructions

Every construction here builds an active-entity memory, and they differ only in what the extraction step may condition on. Text only reads a chunk on its own, Text + History adds the hierarchical history, Text + Previous Memory adds the recalled records, and Full SAADI adds both. Query-time scoring is identical throughout, so any difference in retrieval comes from the construction.

Table 12 reports the outcome. Full SAADI is at or above Text only in all sixteen cells, strictly above in fifteen of them and level on LooGLE under Contriever at HR@10. History-only conditioning is stronger than memory-only in fourteen of the sixteen, and both exceptions fall on LooGLE under Contriever. Recalled records on their own sit at or below Text only in thirteen of the sixteen. Added to the history, they are at or above history-only conditioning in ten of the sixteen cells and below it in six, by 0.2 to 1.3 points, three of them on MuSiQue-32K. The gain from conditioning therefore comes from the history, and neither source dominates every cell. Whether a constructed state helps at all is a separate question, answered by the component ablation in Section 5.

D SAADI Evaluations

This appendix examines the state that the index-time pass builds. The first part asks whether that state is faithful to the document, checking the entity records and the hierarchical summaries against the text they were derived from. The second part describes its size and shape, how many records a document produces, how long the summaries are at each level, and how both accumulate as a document is read. The original chunks are untouched throughout, so nothing here evaluates a rewritten passage. Results cover LooGLE, MuSiQue-32K, QASPER-64K and NovelHopQA.

Table 12: Ablation of the information available when active entities are extracted during index-time construction, with † marking the default. Text only uses the current chunk, Text + History adds the hierarchical document history, Text + Previous Memory adds the recalled active-entity records, and Full SAADI conditions on both. Query-time scoring is identical across the four constructions, and all values use BM25 and Contriever at the fixed cutoffs. Best and second-best values are marked within each dataset block of a column.

Dataset	State conditioning	HR@5		HR@10	
		BM25	Contriever	BM25	Contriever
LooGLE	Text only	91.9	**74.9**	96.1	**85.7**
	Text + History	**92.5**	74.2	**96.2**	85.4
	Text + Previous Memory	91.3	**74.9**	96.0	**86.0**
	Full SAADI†	**92.3**	**75.4**	**96.5**	**85.7**
MuSiQue-32K	Text only	60.1	65.2	75.6	80.2
	Text + History	**64.1**	**69.4**	**78.6**	**83.6**
	Text + Previous Memory	59.0	64.7	74.9	79.8
	Full SAADI†	**62.8**	**68.8**	**77.7**	**83.7**
QASPER-64K	Text only	**15.2**	**15.7**	23.3	24.5
	Text + History	**17.0**	**16.2**	**25.8**	**26.0**
	Text + Previous Memory	**15.2**	14.7	23.7	23.6
	Full SAADI†	**17.0**	**16.2**	**26.1**	**25.8**
NovelHopQA	Text only	17.5	12.2	**21.4**	17.7
	Text + History	**17.8**	**12.9**	**22.0**	**18.8**
	Text + Previous Memory	17.1	12.4	21.3	17.7
	Full SAADI†	**18.1**	**13.5**	**22.0**	**18.2**

Table 13: Automatic evaluation of active-entity faithfulness under the default construction. Each sampled record is checked against the chunk it was extracted from, Faithful overall is the share satisfying all three criteria, and Avg. score is the judge's mean rating out of five. The judge is the construction model itself and no human annotation is involved, so the figures describe agreement with an automatic assessment.

Dataset	Reviewed (N)	Evaluation criterion satisfied (%)			Faithful overall (%)	Avg. score
		Grounded names	Supported aliases	Supported descriptions		
LooGLE	100	98.0	99.0	96.0	96.0	4.86
MuSiQue-32K	100	94.0	91.0	90.0	87.0	4.51
QASPER-64K	100	99.0	100.0	98.0	98.0	4.93
NovelHopQA	100	92.0	91.0	89.0	89.0	4.58

D.1 Faithfulness

Both evaluations below are automatic. The judge is the construction model itself, Qwen3-30B-A3B. It was shown each sampled item together with the text it came from and asked to judge it against fixed criteria, so the numbers report a model's assessment of its own output rather than a human annotator's.

D.1.1 Active-Entity Faithfulness

One hundred active entities were sampled per dataset, and each was checked on three counts, whether its name is grounded in the chunk it was extracted from, whether its aliases are supported there, and whether its descriptions are. Table 13 reports the three rates separately, the share of records satisfying all of them, and a mean score. Names are grounded in 92 to 99 percent of the sampled records and aliases supported in 91 to 100 percent. Descriptions, the weakest field, are supported in 89 to 98 percent, since a description asserts something about an entity instead of pointing at a surface form. All three criteria hold for 87 to 98 percent of records.

D.1.2 Hierarchical-Summary Faithfulness

The same procedure was applied to the summaries, sampling up to one hundred per dataset and asking whether each is supported by the units it was folded from. Table 14 separates unsupported content into hallucination, where a summary adds material that its inputs do not contain, and distortion,

Table 14: Automatic evaluation of hierarchical-summary faithfulness under the default construction. Each sampled summary is judged against the units it was folded from, and a faithful summary carries no unsupported additions and no changed claim. Avg. score is the judge's mean rating out of five, and the judge is the construction model itself, so the figures describe agreement with an automatic assessment rather than a human one.

Dataset	Reviewed (N)	Summary assessment (%)			Avg. score
		Faithful	With hallucination	With distortion	
LooGLE	100	95.0	5.0	0.0	4.81
MuSiQue-32K	100	99.0	1.0	0.0	4.96
QASPER-64K	100	100.0	0.0	0.0	5.00
NovelHopQA	100	99.0	1.0	0.0	4.96

Table 15: Document-level statistics of the active-entity state, averaged over complete documents. AE is the number of unique active entities, and Active chunks counts the chunks carrying at least one of them. Table 16 reports the same statistics as documents are expanded.

Dataset	Chunks/doc.	Active chunks/doc.	AE/doc.	Aliases/doc.	Descriptions/doc.
LooGLE	31.86	30.63	150.24	192.68	273.58
MuSiQue-32K	380.04	377.56	1417.38	1901.16	2948.87
QASPER-64K	746.04	742.39	2869.91	4531.65	5111.91
NovelHopQA	429.78	427.44	1809.44	3322.44	4841.33

where it changes material they do. Distortion does not appear in the sample at all, and hallucination is found in five of the one hundred LooGLE summaries and in at most one per hundred on the other three datasets.

D.2 State Statistics and Structure

D.2.1 Active-Entity Statistics

Table 15 reports the size of the state a document produces. Between 96 and 99.5 percent of chunks carry at least one entity, the state holds 3.7 to 4.7 records per chunk, and each record carries 1.3 to 1.8 aliases and 1.8 to 2.7 descriptions. Table 16 repeats the measurement as documents are extended, where the state grows in proportion to the number of chunks rather than faster, at 3.6 to 3.9 records per chunk at every expansion.

D.2.2 Hierarchical-Summary Statistics

Table 17 reports how many summaries a document receives and how long they are at each level, and its counts are what Appendix B.5.1 checks its predicted call count against. Summary length stays between 38 and 69 tokens at every level, so a high-level summary standing in for hundreds of chunks is about as long as a level-one summary standing in for f of them. That is the compression the bounded history of Appendix B.3 relies on. Figure 9 tracks the total token length of the hierarchical summaries retained after each chunk, excluding raw chunks. Its sawtooth structure reflects the creation of new summaries and their periodic replacement by higher-level folds.

D.2.3 Cumulative Growth

Figure 10 follows the state over normalized document position, separating entity names from the aliases and descriptions attached to them. All three accumulate steadily rather than levelling off, so later parts of a document keep contributing new material instead of repeating what was established in the opening.

E Experimental Setup Details

This appendix records the details behind Section 4.

Table 16: Growth of the active-entity state as documents are expanded, with columns as in Table 15 and averages taken over complete documents. The main results use MuSiQue at 32K and QASPER at 64K, the two rows that also appear in that table.

Dataset	Expansion	Chunks/doc.	Active chunks/doc.	AE/doc.	Aliases/doc.	Descriptions/doc.
MuSiQue	No expansion	19.29	19.04	68.45	74.19	128.93
	32K	380.04	377.56	1417.38	1901.16	2948.87
	64K	743.00	739.30	2807.57	3941.57	5943.43
	128K	1424.25	1417.83	5403.00	7364.83	11552.00
	256K	2848.00	2835.83	10924.83	16827.67	24251.50
QASPER	No expansion	38.45	38.22	148.61	190.33	234.37
	32K	379.87	378.02	1456.93	2202.76	2559.59
	64K	746.04	742.39	2869.91	4531.65	5111.91
	128K	1444.18	1438.73	5584.73	9207.73	9717.09
	256K	2647.67	2634.67	10164.83	17299.83	20005.33

Table 17: Construction statistics of the hierarchical history at $f = 5$. Counts are means over the corpus, while the length at each level is averaged over the documents that reach it, and the highest level is the deepest level any document reaches. A dash marks a level that no document reaches.

Dataset	Summaries/doc.	Mean summary length (tokens)				
		H_1	H_2	H_3	H_4	Highest level
LooGLE	6.67	48.7	44.5	—	—	H_2
MuSiQue-32K	93.16	69.2	65.6	58.5	—	H_3
QASPER-64K	184.57	41.1	42.1	41.6	38.4	H_4
NovelHopQA	105.44	46.8	45.0	44.5	48.0	H_4

E.1 Datasets and Query Populations

Datasets and conditions. We evaluate on LooGLE, MuSiQue, QASPER and NovelHopQA. LooGLE and NovelHopQA are used in their native condition, while MuSiQue and QASPER are expanded and the main results use the 32K and 64K targets. A length suffix such as MuSiQue-32K names that target rather than the measured length of the condition, and Table 16 reports the chunk counts each condition produces.

Query populations. Within each dataset and condition, every method is scored on the same query set: 1,832 queries on LooGLE, 900 on MuSiQue, 1,333 on QASPER, and 908 on NovelHopQA. A query with no annotated evidence cannot be projected onto chunks and is therefore outside the evaluation. MuSiQue is evaluated with a subset of the multi-hop query set, which carries an equal number of 300 queries at two, three, and four hops, and LooGLE, QASPER, and NovelHopQA are evaluated on their full released query sets under the same rule. For MuSiQue and QASPER, the same population is used at every expansion target, so the expansion results are comparable across lengths.

E.2 Chunking and Context Expansion

Chunking. Chunks accumulate consecutive sentences up to a fixed token length, so a boundary is always a sentence boundary and no text is shared between neighbours. The length is 100 tokens for MuSiQue and QASPER and 500 tokens for LooGLE and NovelHopQA. Every method retrieves from the same units, and segmentation is never varied within a dataset.

Why the chunk length differs by source. The unit size follows the source rather than the query. MuSiQue and QASPER are assembled from short self-contained passages, for which chunks near 100 tokens are the range conventionally used, while LooGLE and NovelHopQA are narrative sources, conventionally segmented several times more coarsely (Bhat et al., 2025; Shaukat et al., 2026; Bennani & Moslonka, 2026). The choice also keeps the measurement informative. A native MuSiQue document is short enough that 500-token chunks would leave only a handful of them, so a result at depth ten would return most of the document and the hit rate would stop discriminating

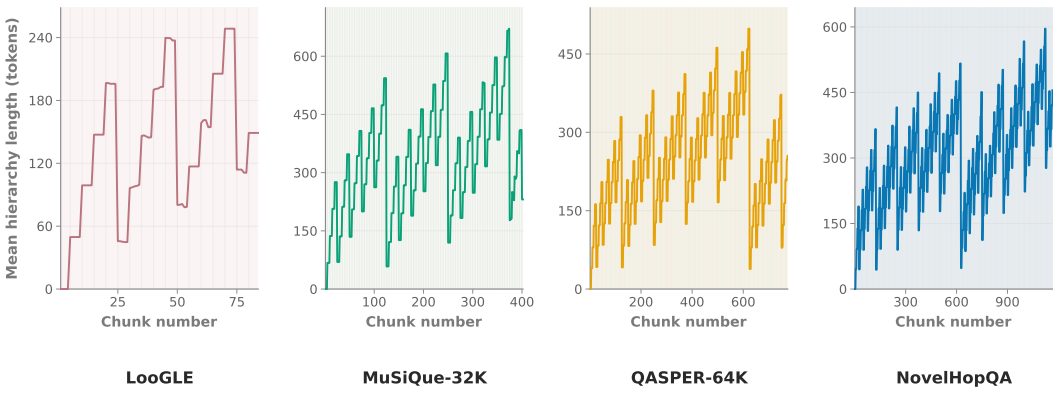

Figure 9: Evolution of the retained hierarchical-summary context during document processing. Each panel reports the mean total summary-token length after each chunk; raw chunks are excluded, and vertical guides mark every fifth step, where $f = 5$ permits a fold.

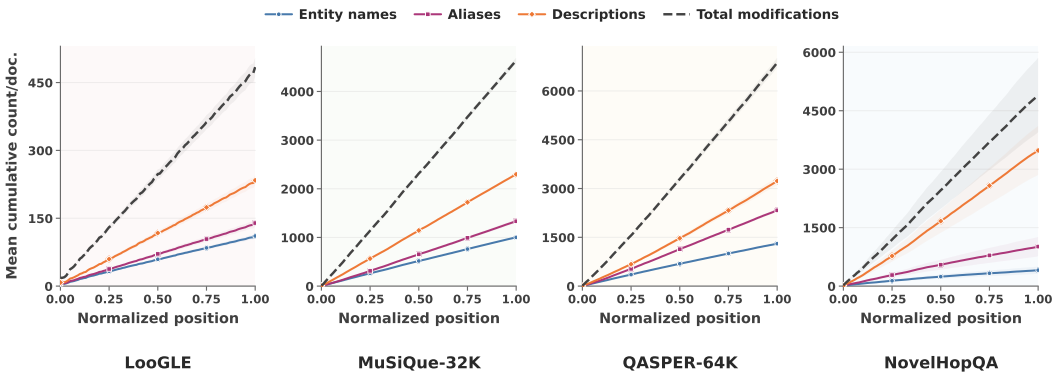

Figure 10: Cumulative growth of SAADI's active-entity state over normalized document position. Panels show the mean numbers of unique entity names, aliases, descriptions, and their combined total; shaded regions indicate one standard error of the mean.

between methods. Expansion does not repair this, because the appended distractors are drawn from other documents and share little vocabulary with the original, which leaves the evidence-bearing chunks separable on lexical overlap alone. Retrieval in the no-expansion condition is already close to ceiling for every method, and a coarser unit would put the remaining conditions there as well.

Context expansion. MuSiQue and QASPER are expanded by grouping their documents rather than by drawing separate distractors for each one. Documents are shuffled and assembled into groups, and a group grows by whole documents until it first reaches the target of 32K, 64K, 128K or 256K tokens. Every document in a group keeps its own queries, so within that group it is the evidence-bearing document for those queries and a distractor for the queries of the others. No document is cut, since a truncated one would lose the evidence its own queries depend on, and the measured length of a group therefore runs above the nominal target. Table 16 reports the chunk counts each condition produces. The query, the reference spans, and the evidence-bearing passages are carried over unchanged, so a query has the same target in every condition, and only the surrounding material grows. LooGLE and NovelHopQA are used in their native condition.

E.3 Evidence Mapping and Metrics

Projection onto chunks. Each dataset annotates the evidence a query depends on as one or more spans of the source document, with one span per hop for a multi-hop query. We project every span onto the chunk sequence and take the union of the chunks they occupy, which is the target of the query and the set a single retrieval would have to return for it to be fully supported. A target holds

Table 18: Retrieval backends, with the checkpoint or the parameters used for each.

Backend	Checkpoint or parameters	Pooling
BM25	$k_1 = 1.5, b_{\mathrm{BM25}} = 0.75$	not applicable
Contriever	`facebook/contriever`	mean
Jina	`jinaai/jina-embeddings-v3`	mean
Qwen	`Qwen/Qwen3-Embedding-8B`	model default
BGE-M3	`BAAI/bge-m3`	CLS

one chunk when the annotated evidence sits inside one, and several when the evidence is distributed, whether because a span crosses a chunk boundary or because a multi-hop question draws on separate passages.

An example. Consider a document in which C_7 states the name of a person's father, while C_1 and C_2 together state that father's occupation, the statement being long enough that the segmentation splits it across the two chunks. A question asking for the father's name is answered by C_7 alone, so its target is $\{C_7\}$. A question asking for the father's occupation needs the name before the occupation can be attributed to the right person, so its target is $\{C_1, C_2, C_7\}$, satisfied only when all three chunks are returned in the same pass. The second question can never be satisfied at a depth below three, whatever the ranking.

The measure. A query is satisfied at depth k when the returned set contains its complete target, and HR@k is the satisfied fraction of the dataset,

$$\mathrm{HR@}k \;=\; \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathbf{1}\big[\, \mathcal{E}_q \subseteq \mathcal{R}_k(q) \,\big], \tag{14}$$

where $\mathcal{Q}$ is the query population of the dataset, $\mathcal{E}_q$ the target of query q, and $\mathcal{R}_k(q)$ the set returned for it at depth k. Every method is scored on the same population at the same depth.

Why the measure is strict. A target larger than k cannot be returned in full at any depth below its size, so a query whose evidence is spread over more chunks than the depth allows counts as a miss however well it is ranked. The effect is sharpest at $k = 5$ on NovelHopQA, whose queries carry up to four hops; a four-hop target already occupies four of the five positions, and a span that crosses a chunk boundary occupies the fifth, so the query is satisfied only by an exact top-5. The behaviour belongs to the measure and the data rather than to any system, and it applies identically to every method compared here.

E.4 Retrievers

Table 18 lists the backends: BM25 (Robertson & Zaragoza, 2009), Contriever (Izacard et al., 2021), `jina-embeddings-v3` (Sturua et al., 2024), `Qwen3-Embedding-8B` (Zhang et al., 2025b), and `bge-m3` (Chen et al., 2024). All dense encoders are used off the shelf without fine-tuning, and ablations use BM25 unless a result states otherwise.

E.5 SAADI Configuration

Construction. Index-time state is built with `Qwen/Qwen3-30B-A3B-Instruct-2507` (Yang et al., 2025a) at revision `main`, checkpoint `0d7cf23991f47feeb3a57ecb4c9cee8ea4a17bfe`, which performs both the state update of Equation (1) and the fold of Equation (5) without fine-tuning. Decoding is greedy, at temperature zero. The folding factor is $f = 5$. Selective recall scores the incoming chunk against the recorded text with BM25 and returns at most five records, each with at most two aliases and three descriptions. Sensitivity to the construction model, the folding factor, the recall budget, and the selector is reported in Appendix C.

Fusion weights. The channel weights are $(\lambda_T, \lambda_M, \lambda_S) = (1, 0.5, 0.3)$ and the field weights $(\mu_N, \mu_A, \mu_D) = (1, 1, 1)$, applied after z-score standardization of each channel over the candidate

Table 19: Run-to-run variation of SAADI under BM25. State construction with the default model was repeated three times independently at temperature zero, and each cell reports the mean and sample standard deviation over the three runs. All other tables in the paper report the final run.

Dataset	HR@5	HR@10
LooGLE	92.3 ± 0.2	96.4 ± 0.3
MuSiQue-32K	62.8 ± 0.4	77.7 ± 0.2
QASPER-64K	17.1 ± 0.1	26.2 ± 0.2
NovelHopQA	18.1 ± 0.0	21.9 ± 0.2

chunks. They are selected once, on a fixed random sample of 100 queries drawn from the LooGLE and NovelHopQA evaluation sets, and then held constant for every dataset, backend, depth, and condition. The sample is not held out from the results reported for those two datasets. They are not retuned per dataset. MuSiQue and QASPER take no part in the selection, so the numbers reported for those two datasets are obtained under weights chosen without reference to them. Sensitivity is reported in Appendix C.5.

Repeated construction. State construction with the default model was run three times independently at temperature zero, and Table 19 reports the mean and standard deviation of every hit rate under BM25. The largest standard deviation in any cell is 0.4 points, on MuSiQue-32K at HR@5, and the largest range over the three runs is 0.7 points. The smallest margin the body relies on under BM25 is SAADI's 1.1-point lead over Vanilla on LooGLE at HR@10, which is more than three times the standard deviation of that cell. All tables report the final run, which was not selected on score.

E.6 Baselines and Evidence Accounting

Fixed-depth baselines. Vanilla retrieval, Late Chunking, LlamaIndex, LQCA, HippoRAG v2, and RAPTOR are evaluated at $k \in \{5, 10\}$ over the same chunk sequence as SAADI. Three of them need a stated convention, because their retrieval units or their configuration are not fixed by the method itself, and one needs a note on its configuration.

HippoRAG v2. HippoRAG v2 is run as released, in its default configuration, including the model call that filters retrieved triples at query time. It is the one fixed-depth baseline that makes a model call at query time; the fixed-depth comparison matches methods at depth, so that call does not enter it.

Matching the LlamaIndex granularity. LlamaIndex segments semantically and takes a similarity threshold rather than a chunk size, so its units are not fixed in advance and one threshold yields different granularities on different corpora. For each dataset we sweep the threshold and keep the value whose resulting mean chunk size is closest to the fixed chunk length used everywhere else. The comparison therefore holds granularity approximately constant and isolates the segmentation strategy rather than the unit size.

Late Chunking. Late Chunking pools each chunk embedding over a surrounding window of a long-context encoding, and how far that window overlaps neighbouring chunks is a free parameter. It requires a long-context embedding model, and in our grid it is available under `jina-embeddings-v3` only, so it cannot be run under BM25 or the other encoders. We run it at three overlaps, 0, 25, and 50 tokens for the 100-token chunking of MuSiQue and QASPER and 0, 50, and 100 tokens for the 500-token chunking of LooGLE and NovelHopQA. All three appear as separate rows of the Jina block of Table 1. This favours the baseline in the same direction as the RAPTOR conversion below, so its ranked values should be read as the best configuration we tried rather than a single tuned one.

Scoring RAPTOR. RAPTOR retrieves over a flattened tree whose units are either leaves, each carrying one chunk, or internal nodes, each carrying a summary of several. Comparing it at a fixed depth requires a conversion, and we adopt the one most favourable to the method. A retrieved unit

is charged the number of chunks it covers and credited with all of them, so a node summarizing four chunks consumes four of the k slots and counts as having returned each of the four. When the next unit in rank order would exceed the remaining slots, it is admitted anyway if it covers a target chunk, so the depth never prevents RAPTOR from being credited with evidence its ranking reached. A leaf is never returned twice, and since the ordering is already resolved in RAPTOR's favour, overlap between internal nodes does not change whether a target is credited. Charging by coverage is the conservative half of this conversion and crediting by coverage is the generous half, and the second dominates, since a summary is treated as conveying every chunk beneath it without loss. The resulting rows are a ceiling on what RAPTOR could reach at a given depth rather than an operating point.

A RAPTOR example. Take $k = 5$. A ranking that returns the leaf C_1, then a node summarizing C_{10}, C_{11}, and C_{12}, then the leaf C_8 has consumed exactly five slots with three retrieved units, and is credited with $\{C_1, C_8, C_{10}, C_{11}, C_{12}\}$, so a query whose target is C_{12} counts as satisfied. Suppose instead that four leaves have already filled four slots and the next unit is a node covering C_1 through C_4, which would need four more. If the target lies inside that node, say C_3, the node is admitted regardless and the query is still credited. The second case is where the conversion becomes generous, and it is the reason these rows are read as a ceiling rather than as a measurement.

Agentic baselines. RAG-R1, Agentic-R, A-RAG, GRIP, and UMA choose how much of a document to read, so no depth describes them. Each is run as released; its policy model, prompts, decoding settings, search tool, and stopping rule are unchanged, and its search tool indexes the same chunk sequence as every other method, restricted to the document or expanded group attached to the query. We intervene only to log the trajectory, which is cut at the agent's own answer boundary, the first token of answer generation. Each is credited with its pre-answer exposure, the first-seen deduplicated set of chunks whose content entered a model call before final-answer generation. Candidates that a retrieval or memory tool scores internally are not counted unless their text is passed to the model, and a chunk shown to any intermediate call is counted even if it is not retained afterwards. For A-RAG, which returns search-result previews before a full read, only explicit full-chunk reads are counted, and previews are excluded. Final-answer generation is excluded from every count, so the comparison concerns retrieval rather than answer quality. In the comparison of Appendix F.4, SAADI is evaluated under BM25 at a fixed ladder of depths from 3 to 20, so that a row exists at or below the mean exposure of every agent on every dataset, and each agent is also read against the integer depth nearest its own mean exposure. In the expansion and hop breakdowns, it is read at the two integer depths bracketing that exposure in each cell, so that no comparison there depends on rounding in one direction.

Resource accounting. For each agentic method we record model calls, input tokens, and generated tokens over the complete pre-answer trajectory, read from backend usage. Input tokens are counted per call, so context that an agent re-sends on later turns is counted each time it is sent. The crossover of Section 5 is the number of queries per document at which that cumulative per-query total meets SAADI's one-time construction cost, which instantiates the break-even of Section 3.3.

Automatic judging. The faithfulness checks of Appendix D.1 are produced by the construction model, applied to each record or summary together with the source text it was derived from. The procedure involves no human annotation, so the results describe agreement with an automatic assessment.

F Results Details

F.1 The full retrieval grid

Which methods the grid covers. The grid of Table 1 holds Vanilla, LateChunking, LlamaIndex, LQCA, and SAADI. Four of them take any retrieval backend, so each can be run over all five retrievers, and the grid can separate what a method contributes from what its backend contributes. Late Chunking is the exception within the grid. It depends on a long-context embedding model, so it is run under `jina-embeddings-v3`, the one retriever in the grid that supports it, and cannot be run under BM25 or the other encoders. Its three overlap settings appear as separate rows of the Jina

Table 20: Leave-one-dataset-out backend selection on HR@10. For each held-out dataset, each method selects its backend using only the remaining datasets; each cell reports the held-out score, rank, and selected backend. The summary reports mean rank, number of first-place results, worst rank, and worst selection regret across held-out datasets.Ties in mean rank are broken by the order in which backends are listed in Table 18.

Method	Held-out Results				Overall Summary			
	LooGLE	MuSiQue-32K	QASPER-64K	NovelHopQA	Mean ↓	#Wins ↑	Worst ↓	Regret ↓
SAADI	**90.1** (#1) BGE-M3	**88.3** (#2) BGE-M3	**30.9** (#2) BGE-M3	**20.6** (#1) Qwen	**1.50**	**2**	**2**	**6.40**
LQCA	86.8 (#3) Qwen	**90.4** (#1) Qwen	**53.6** (#1) Qwen	1.9 (#4) BM25	**2.25**	**2**	4	**7.40**
LlamaIndex	**87.3** (#2) BGE-M3	82.4 (#4) BGE-M3	30.3 (#3) BGE-M3	18.2 (#3) BGE-M3	3.00	0	4	8.20
Vanilla	85.9 (#4) Qwen	87.4 (#3) BGE-M3	29.3 (#4) BGE-M3	**19.8** (#2) Qwen	3.25	0	4	9.50

block, and the ranking uses the best of them in each block (Appendix F.11). The remaining fixed-depth baselines do not enter the grid. HippoRAG v2 builds and queries its own retrieval stack, so exchanging it for BM25 or a sentence encoder would no longer be the same method, and it is compared with SAADI at matched depth in Appendix F.4. RAPTOR retrieves summary nodes alongside passages and is scored under a conversion that resolves every choice in its favour, so its rows are an upper bound rather than an operating point and are read as such.

How the mean rank is computed. Mean ranks over the grid are computed from these $5 \times 4 \times 2 = 40$ cells, one for each combination of dataset, retriever, and depth, and they are the basis for the count reported in Section 5.1. Within a cell the methods present are ranked by hit rate, five in the Jina block and four elsewhere, with higher values ranked better and ties assigned their average rank. The reported figure is the arithmetic mean of a method's cell ranks, taken over all 40 cells for the four methods run under every backend and over the 8 cells of the Jina block for Late Chunking, which enters each cell with its best overlap setting. At HR@5 and HR@10 the mean ranks are 1.78 and 1.65 for SAADI, 2.35 and 2.33 for LQCA, 2.70 and 2.90 for Vanilla, 3.33 and 3.33 for LLAMAINDEX, and 4.25 and 4.00 for Late Chunking, and SAADI ranks first or second in 34 of the 40 cells. The figure measures consistency across datasets, backends, and depths rather than performance in any one configuration.

F.2 CROSS-DATASET BACKEND SELECTION

The full retrieval grid allows each method to be paired with several retrieval backends, but selecting the best backend on the target dataset would use information that is unavailable in deployment. We therefore evaluate backend transfer in a leave-one-dataset-out setting. For each method and held-out dataset, we rank its candidate backends separately on each of the other three datasets using HR@10, assign average ranks to ties, and select the backend with the lowest mean rank across those three datasets. Ties in mean rank are broken by the order in which the backends are listed in Table 18. Rank-based selection gives each source dataset equal influence regardless of differences in the scale or spread of its hit rates across backends. The selected backend is then fixed and evaluated on the held-out dataset, and the procedure is repeated independently for all four held-out datasets. It is applied to the four methods present under every backend. Late Chunking is excluded because it is available under one backend only, so no cross-backend selection exists for it.

Table 20 reports the held-out HR@10, the resulting method rank, and the backend selected in each split. The summary columns report mean held-out rank, the number of first-place results (#Wins), the worst observed rank (Worst), and worst-case selection regret (Regret). For method m and held-out dataset d, we define

$$\mathrm{Regret}_{m,d} = S^{\mathrm{oracle}}_{m,d} - S^{\mathrm{selected}}_{m,d},$$

where $S^{\mathrm{oracle}}_{m,d}$ is the best HR@10 attained by any backend for method m on dataset d, and $S^{\mathrm{selected}}_{m,d}$ is the score obtained by the backend chosen without using d. The reported Regret is $\max_d \mathrm{Regret}_{m,d}$, measured in percentage points. Because this is an absolute difference, its magnitude should be interpreted with the score range of the corresponding method in mind.

SAADI attains a mean held-out rank of 1.50, ranks first on two of the four held-out datasets, never falls below second, and has a worst-case selection regret of 6.40 percentage points. That regret comes from LooGLE, where the held-out procedure selects BGE-M3 while BM25 is the oracle

Table 21: Hop-stratified retrieval on MuSiQue and NovelHopQA under BM25, with SAADI reported at several cutoffs and the agentic methods scored on the chunks they admit before answering, so they carry no cutoff. The Overall column is the query-weighted mean over the hop bins, which are equally sized on MuSiQue and unequal on NovelHopQA. Values are percentages.

Approach	Cutoff	Overall		1-hop		2-hop		3-hop		4-hop	
		Avg.	HR	Avg.	HR	Avg.	HR	Avg.	HR	Avg.	HR
MuSiQue-32K											
RAG-R1	Trajectory	8.09	86.0	–	–	6.22	83.0	8.43	83.7	9.61	91.3
Agentic-R	Trajectory	5.54	75.2	–	–	4.64	73.0	5.58	72.7	6.41	80.0
A-RAG	Trajectory	2.66	45.1	–	–	2.01	46.3	2.86	35.7	3.12	53.3
GRIP	Trajectory	3.99	50.7	–	–	3.59	45.3	4.00	49.3	4.37	57.3
UMA	Trajectory	19.59	**90.3**	–	–	17.62	**89.0**	19.47	**87.7**	21.67	**94.3**
BM25 Vanilla	10	10.00	74.0	–	–	10.00	71.3	10.00	71.0	10.00	79.7
SAADI(BM25)	3	3.00	49.2	–	–	3.00	48.0	3.00	47.0	3.00	52.7
	4	4.00	57.6	–	–	4.00	53.0	4.00	56.3	4.00	63.3
	5	5.00	62.8	–	–	5.00	58.3	5.00	60.0	5.00	70.0
	10	10.00	77.7	–	–	10.00	76.3	10.00	75.0	10.00	81.7
	20	20.00	**91.9**	–	–	20.00	**92.3**	20.00	**90.3**	20.00	**93.0**
NovelHopQA											
RAG-R1	Trajectory	5.93	14.1	5.63	12.7	6.05	20.5	6.20	16.0	6.16	9.4
Agentic-R	Trajectory	3.91	12.6	3.67	11.4	4.03	15.8	4.02	14.8	4.17	9.9
A-RAG	Trajectory	3.89	9.0	3.64	15.1	4.06	7.0	4.08	3.6	4.03	3.7
GRIP	Trajectory	4.95	14.2	4.89	18.8	5.14	16.4	5.14	10.1	4.75	6.8
UMA	Trajectory	3.72	14.0	3.42	18.6	3.79	15.8	3.98	11.8	4.03	5.2
BM25 Vanilla	10	10.00	**20.0**	10.00	**25.2**	10.00	21.1	10.00	**18.3**	10.00	**10.5**
SAADI(BM25)	3	3.00	15.4	3.00	20.2	3.00	17.5	3.00	12.4	3.00	6.8
	4	4.00	16.7	4.00	21.2	4.00	20.5	4.00	14.2	4.00	6.8
	5	5.00	18.1	5.00	22.3	5.00	**21.6**	5.00	15.4	5.00	8.9
	10	10.00	**22.0**	10.00	**26.0**	10.00	**26.9**	10.00	**20.7**	10.00	**11.0**

choice. LQCA also records two first-place results but reaches a mean rank of 2.25, a worst rank of 4, and 7.40 points of worst-case regret. The corresponding mean-rank, worst-rank, and regret values are 3.00, 4, and 8.20 for LLAMAINDEX, and 3.25, 4, and 9.50 for Vanilla. With four held-out splits, we treat these results as evidence of cross-dataset stability rather than as a high-precision estimate of generalization.

F.3 MULTI-HOP STRATIFICATION

Table 21 stratifies retrieval by the number of evidence hops on NovelHopQA and MuSiQue-32K. NovelHopQA carries one- through four-hop queries, and MuSiQue-32K carries 300 queries at each of two, three, and four hops with no single-hop queries. Fixed-depth methods are evaluated at $k \in \{5, 10\}$, and agentic methods are credited with the complete first-seen deduplicated set of chunks exposed before answer generation, following the accounting protocol of Appendix E.

SAADI is ahead of Vanilla at HR@10 in every hop level on both datasets. Under BM25 at $k = 10$ the NovelHopQA groups move from 21.1 to 26.9, 18.3 to 20.7, and 10.5 to 11.0 across two, three, and four hops, and the MuSiQue-32K groups move from 71.3 to 76.3, 71.0 to 75.0, and 79.7 to 81.7. The gain survives at higher hop counts, though its size is not monotonic in the number of hops. The narrowest case is four-hop NovelHopQA, where SAADI reaches 11.0 against 10.5 for Vanilla, as noted in Section 5.1.

The hop trend differs between the two datasets once the agentic rows are read alongside the fixed-depth ones. On NovelHopQA the best cell at every hop level is a SAADI row, and the agentic methods stay below both fixed-depth methods throughout, with RAG-R1 at 9.4 at four hops against 11.0 for SAADI. On MuSiQue-32K the agentic methods gain as hops increase, and at four hops RAG-R1 reaches 91.3 on a trajectory of 9.61 chunks, above SAADI at depth 10, which reaches 81.7, and below SAADI at depth 20, which reaches 93.0. RAG-R1 is therefore ahead of SAADI at depth 10, the depth nearest its exposure, and behind it at depth 20.

Table 22: Retrieval hit rate for fixed-depth and agentic methods on the shared per-dataset evaluation set. Agentic rows are scored on the deduplicated chunks whose content entered the trajectory, so Avg. chunks is the observed mean exposure there and the stated cutoff for fixed-depth rows. SAADI is reported under BM25 at cutoffs 3 to 20. Best and second-best values are marked within each HR column.

Method	Cutoff	LooGLE		MuSiQue-32K		QASPER-64K		NovelHopQA	
		Avg. chunks	HR	Avg. chunks	HR	Avg. chunks	HR	Avg. chunks	HR
HippoRAG v2	5	5.00	84.90	5.00	67.00	5.00	15.80	5.00	13.90
	10	10.00	91.80	10.00	78.30	10.00	24.30	10.00	18.10
RAPTOR (Upper Bound)	5	5.00	62.30	5.00	54.70	5.00	11.90	5.00	7.30
	10	10.00	72.20	10.00	68.00	10.00	19.30	10.00	10.10
RAG-R1	Trajectory	4.49	71.07	8.09	86.00	7.64	19.95	5.93	14.10
Agentic-R	Trajectory	3.37	54.09	5.54	75.22	3.76	16.13	3.91	12.56
A-RAG	Trajectory	1.95	68.56	2.66	45.11	3.48	12.60	3.89	9.03
GRIP	Trajectory	4.57	86.74	3.99	50.67	3.91	10.28	4.95	14.21
UMA	Trajectory	3.06	87.83	19.59	**90.33**	14.79	27.31	3.72	13.99
SAADI (BM25)	3	3.00	87.94	3.00	49.22	3.00	11.70	3.00	15.42
	4	4.00	90.56	4.00	57.56	4.00	14.33	4.00	16.74
	5	5.00	92.30	5.00	62.78	5.00	16.95	5.00	18.06
	10	10.00	96.45	10.00	77.67	10.00	26.11	10.00	22.03
	15	15.00	**98.31**	15.00	88.00	15.00	**32.33**	15.00	**24.34**
	20	20.00	**99.13**	20.00	**91.89**	20.00	**36.31**	20.00	**26.76**

F.4 Agentic comparison, exact scores and exposure

Figure 2 summarizes retrieval quality as a function of pre-answer evidence exposure, and Table 22 reports the exact values behind it. For fixed-depth methods, exposure is the retrieval depth, and for agentic methods it is the mean number of first-seen distinct raw chunks whose content enters the trajectory before answer generation. The second quantity is an observed average rather than a depth the method holds to, so the comparison matches the amount of evidence made available rather than the number of model calls.

SAADI is reported under BM25 at depths from 3 to 20 so that a fixed-depth row exists at or below the mean exposure of every agentic method on every dataset, including UMA at 19.59 chunks on MuSiQue-32K. BM25 is the paper's default backend and is used throughout this comparison. It is not the backend the held-out procedure of Appendix F.2 would select, which is BGE-M3 on three of the four datasets and gives SAADI higher values on MuSiQue-32K and QASPER-64K, so the BM25 operating points reported here are the more conservative of the two. Table 23 gives SAADI under BM25 at every depth and is the source of the SAADI curve in Figure 2.

The ordering among the agentic methods does not carry across corpora. UMA is the strongest of them on LooGLE at 87.8, on MuSiQue-32K at 90.3 and on QASPER-64K at 27.3, while GRIP is the strongest on NovelHopQA at 14.2 and the weakest on QASPER-64K at 10.3. RAG-R1 is second on MuSiQue-32K and QASPER-64K, behind UMA in both, and third on LooGLE. SAADI's ladder is monotone in depth on every dataset, so the depth at which it meets a given agent can be read off the table directly. On MuSiQue-32K it passes RAG-R1 only at depth 15, at 88.0 against 86.0, and on QASPER-64K at depth 7, at 21.4 against 20.0. Among the fixed-depth baselines, HippoRAG v2 at depth 10 is below SAADI at the same depth on three of the four datasets, at 91.8 against 96.5 on LooGLE, 24.3 against 26.1 on QASPER-64K, and 18.1 against 22.0 on NovelHopQA, and above it on MuSiQue-32K, at 78.3 against 77.7. The RAPTOR rows are the upper bound described in Appendix F.11.

Matching at each agent's own exposure. Because the ladder is reported at every integer depth, each agent can also be read against the depth nearest its own mean exposure. SAADI is ahead at that depth in sixteen of the twenty agent and dataset pairs, and at both integer depths bracketing the exposure in fourteen. The four exceptions at the nearest depth fall on MuSiQue-32K, against RAG-R1 and Agentic-R, and on QASPER-64K, against Agentic-R and A-RAG, which are the two corpora where the ladder climbs most slowly over the first few depths. UMA reads the most of the five and

Table 23: SAADI hit rate under BM25 at every retrieval cutoff from one to twenty, with expansion applied throughout. This is the full ladder behind the plotted curves and behind the cutoffs reported in Table 22. Values are percentages.

Top-K	LooGLE HR	MuSiQue-32K HR	QASPER-64K HR	NovelHopQA HR
1	70.09	27.22	4.35	9.47
2	83.08	41.11	8.63	13.33
3	87.94	49.22	11.70	15.42
4	90.56	57.56	14.33	16.74
5	92.30	62.78	16.95	18.06
6	93.56	65.78	19.88	18.72
7	94.43	69.33	21.38	19.82
8	95.47	73.11	23.26	21.04
9	95.91	75.33	24.76	21.15
10	96.45	77.67	26.11	22.03
11	96.72	79.56	27.68	22.25
12	97.27	81.78	29.11	22.58
13	97.60	84.22	30.16	23.24
14	97.87	85.89	31.13	23.46
15	98.31	88.00	32.33	24.34
16	98.42	88.89	33.01	25.00
17	98.74	89.67	33.91	25.44
18	98.91	90.78	35.18	26.21
19	99.02	91.44	35.48	26.54
20	99.13	91.89	36.31	26.76

its exposure varies the most across corpora, from 3.06 chunks per query on LooGLE to 19.59 on MuSiQue-32K, and SAADI is ahead of it at the matched depth on all four datasets, at 87.94 against 87.83 on LooGLE, 91.89 against 90.33 on MuSiQue-32K, 32.33 against 27.31 on QASPER-64K, and 16.74 against 13.99 on NovelHopQA.

Exposure against model calls. Exposure and model calls are separate quantities, since one call can admit several chunks, and UMA is where the two diverge most. It issues between 2.01 and 2.71 calls per query, among the two fewest of any agentic method on every dataset, while its trajectory reaches 19.59 chunks on MuSiQue-32K and 14.79 on QASPER-64K, above every other row by a wide margin. Matching agentic methods by call count would therefore record UMA as one of the cheapest on exactly the two datasets where it reads the most, which is the reason the comparison in this paper is matched on exposure. UMA retrieves from its memory through a tool call, and that tool may score many candidates internally on one call. Only the chunks whose text is passed to the language model are counted, under the same rule applied to every agent, so the trajectory sizes above are the evidence the model saw rather than the number of candidates the tool considered.

Crossovers in tokens. The crossovers of Table 2 count model calls. The same accounting in input tokens and in generated tokens is reported in Table 24. Generated tokens move the crossovers furthest, because construction writes records and summaries while an agent's trajectory mostly reads: in generated tokens the break-even ranges from 256 to 908 queries per document on LooGLE and from $1{,}564$ to $7{,}715$ on QASPER-64K, about an order of magnitude above the call figures.

Comparators that also pay offline. UMA is the one comparator with an offline memory-construction cost, so its crossover uses the offline-plus-online form of Equation 4, $q^*_{m,r} = \max\{(S_r - U_{m,r})/D,\, 0\}/O_{m,r}$, with S_r SAADI's total construction use of resource r, $U_{m,r}$ the comparator's own offline use, D the number of documents, and $O_{m,r}$ the comparator's mean online use per query. The other agents have $U_{m,r} = 0$. On generated tokens UMA has the largest crossover of any comparator on all four datasets, at 908, $2{,}813$, $7{,}715$, and $6{,}692$ queries per document, so SAADI's construction budget takes longer to be repaid against UMA than against any other method here. Its offline term is positive on every dataset for calls, input tokens, and generated

Table 24: Pre-answer LLM resource use per query and resource-equivalent crossover against SAADI's one-time offline index construction per document. RAG-R1, Agentic-R, A-RAG, GRIP, and UMA are evaluated online. For calls, input tokens, and generated tokens, the corresponding crossover is the average queries per document at which cumulative method use equals SAADI's offline construction use. UMA's crossover includes its own offline memory-construction use. Avg. Trajectory Chunks is the mean number of unique chunks admitted to the pre-answer trajectory. For A-RAG, this counts explicit full-chunk reads and excludes search previews. Final-answer generation is excluded.

Dataset	Method	Avg. Trajectory Chunks	LLM Resource Use / Query			SAADI Crossover (Queries/Doc.)		
			Calls	Input Tokens	Generated Tokens	Calls	Input Tokens	Generated Tokens
LooGLE	**RAG-R1**	4.49	2.43	9,864	195	16	15	287
	Agentic-R	3.37	2.65	6,160	219	15	24	256
	A-RAG	1.95	2.25	2,095	141	17	70	398
	GRIP	4.57	2.71	3,681	149	14	40	375
	UMA	3.06	2.01	1,697	61	11	50	908
MuSiQue-32K	**RAG-R1**	8.09	4.15	8,357	339	114	123	782
	Agentic-R	5.54	3.70	3,449	295	128	297	899
	A-RAG	2.66	3.31	3,476	128	143	295	2,074
	GRIP	3.99	2.46	1,109	94	192	924	2,805
	UMA	19.59	2.71	3,980	94	164	225	2,813
QASPER-64K	**RAG-R1**	7.64	3.34	5,447	302	279	346	1,564
	Agentic-R	3.76	2.71	1,699	225	344	1,111	2,102
	A-RAG	3.48	2.03	1,390	94	457	1,358	5,028
	GRIP	3.91	2.40	1,024	80	388	1,844	5,875
	UMA	14.79	2.11	2,010	61	413	818	7,715
NovelHopQA	**RAG-R1**	5.93	2.42	8,775	263	221	235	1,962
	Agentic-R	3.91	2.67	5,657	271	201	364	1,901
	A-RAG	3.89	2.09	1,647	98	256	1,252	5,244
	GRIP	4.95	2.60	3,484	136	206	592	3,791
	UMA	3.72	2.19	2,253	77	169	607	6,692

Table 25: SAADI offline construction calls per document at $f = 5$, averaged over each corpus. State-update calls are one per chunk, and fold calls are the extra summarization calls introduced by hierarchical folding. Totals are measured means, so they differ slightly in either direction from Equation 3 evaluated at the mean chunk count, because folds trigger on each document's own chunk count rather than on the corpus average.

Dataset	Chunks	State-update	Fold	Total
LooGLE	31.86	31.86	6.67	38.53
MuSiQue-32K	380.04	380.04	93.16	473.20
QASPER-64K	746.04	746.04	184.57	930.61
NovelHopQA	429.78	429.78	105.44	535.22

tokens, and its generated-token component is roughly one percent of SAADI's construction total on each dataset, which is consistent with a memory-construction pass that reads widely and writes little.

F.5 SAADI CONSTRUCTION COST BREAKDOWN

SAADI's offline cost separates into one state-update call per chunk and the additional calls introduced by hierarchical folding. Table 25 reports these components at the default $f = 5$. The measured totals are 38.53 calls per document on LooGLE, 473.20 on MuSiQue-32K, 930.61 on QASPER-64K, and 535.22 on NovelHopQA, of which 6.67, 93.16, 184.57, and 105.44 are summary-fold calls (Figure 11). Subtracting the fold calls from each total leaves 31.86, 380.04, 746.04, and 429.78, which are the mean chunk counts of the four corpora, so the per-chunk term is exactly one call per chunk as Section 3.3 states. The totals themselves agree with $T(N, 5)$ evaluated at those chunk counts to within half a percent on all four datasets, so the measured construction cost matches the analysis rather than being fitted to it.

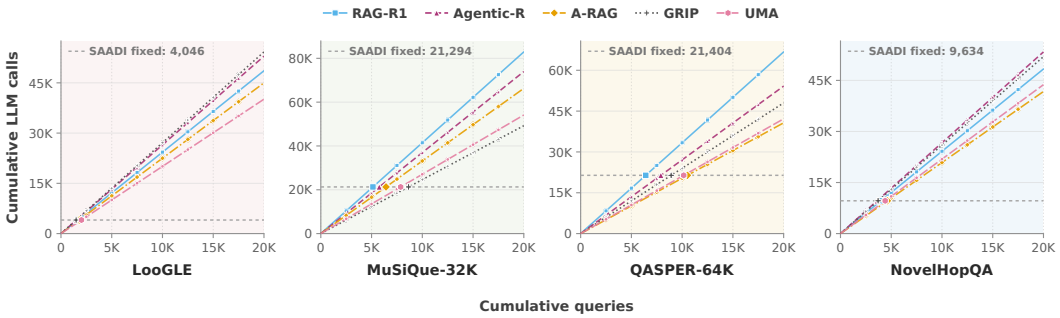

Figure 11: Projected cumulative online retrieval-stage LLM calls over 20K queries. Agent curves multiply each method's average pre-answer calls per query by cumulative queries; final-answer generation and method-side offline construction are excluded. The light dashed gray line is the one-time complete-dataset SAADI construction cost (chunk updating plus hierarchical summarization).

F.6 Retrieval under context expansion

Table 26 reports retrieval at the four expansion targets alongside the unexpanded setting. Vanilla and SAADI appear under BM25 at depths 5, 10, 15, and 20 and under Contriever at depths 5 and 10, alongside the five agents of Section 5.2, and Figure 3(a) plots the depth-10 rows. Every method loses accuracy as distractors are added, so the question is which ordering survives. The reading below compares the depth-10 rows with the four agents whose exposure stays between 2.35 and 9.45 chunks. UMA reads 13.7 to 20.9 chunks per query at these targets and is matched at its own exposure in Appendix F.4. On QASPER the best depth-10 row at every setting is a SAADI row, from 60.6 under Contriever unexpanded to 18.4 under BM25 at 256K, and each of the four is below it throughout. UMA is above the depth-10 rows at three of the five settings, which is consistent with its reading roughly fifteen chunks per query rather than ten. On MuSiQue the ordering changes once expansion begins. SAADI under Contriever is ahead unexpanded at 93.7 against 91.3 for RAG-R1, and RAG-R1 is ahead of every depth-10 row at all four expanded targets, from 86.0 at 32K to 74.7 at 256K. SAADI at depth 20 is above RAG-R1 at every target, at 91.9, 87.3, 82.1, and 74.8, with more than twice its exposure.

The MuSiQue reversal at depth 10 is a change of ordering rather than an artefact of one backend. RAG-R1 leads the depth-10 rows under both BM25 and Contriever at every expanded target, and SAADI regains the lead only at depth 20. Exposure also moves with document length for the agentic methods while the fixed-depth rows stay at their depth. RAG-R1 reads 6.85 chunks per query on unexpanded MuSiQue and 9.45 at 256K, and 5.69 to 8.15 on QASPER, so part of what the agentic trajectory buys at the longer settings is simply more evidence.

F.7 Component ablations

Table 27 reports the two ablations under BM25 at HR@10. Panel (a) varies what the extraction step sees during index-time construction while query-time scoring is held fixed, and panel (b) varies which auxiliary channels enter the fused ranking while construction is held fixed. The Full column is identical in both panels, at 96.5, 77.7, 26.1, and 22.0, since the two panels are independent readings of the same system.

The two behave differently. Construction is not monotone in the amount of state supplied. Hierarchical history improves over no state on all four datasets, at 96.2 against 96.1 on LooGLE, 78.6 against 75.6 on MuSiQue-32K, 25.8 against 23.3 on QASPER-64K, and 22.0 against 21.4 on NovelHopQA. Recalled records supplied without history sit at or below no state on three of the four, and adding them to the history changes HR@10 by -0.9 to $+0.3$; on MuSiQue-32K the history-only setting exceeds the full state, 78.6 against 77.7. At index time, therefore, the records add little beyond the history, which is what lexical recall can be expected to do, since it resolves a surface form only when an earlier record already carries it. Their contribution is at query time, in panel (b).

Fusion is monotone by contrast. Each auxiliary channel improves over text-only retrieval on every dataset, and the full combination is strongest on all four. The largest single-channel gain is the

Table 26: Retrieval under increasing distractor expansion, from no expansion to 256K, with 100-token chunks and zero overlap. Avg. is the mean number of chunks admitted, which can fall below a fixed cutoff on short documents. Vanilla and SAADI use BM25 at cutoffs 5 to 20 and Contriever at 5 and 10, while the agentic methods are scored on their full pre-answer trajectory under the exposure rules of Appendix E. The 32K column for MuSiQue and the 64K column for QASPER are the standard runs reported elsewhere, and best and second-best values are marked within each expansion block. Values are percentages.

Approach	Cutoff	No		32K		64K		128K		256K	
		Avg.	HR	Avg.	HR	Avg.	HR	Avg.	HR	Avg.	HR
MuSiQue											
RAG-R1	Trajectory	6.85	91.3	8.09	86.0	8.55	82.3	8.95	**78.7**	9.45	**74.7**
Agentic-R	Trajectory	5.03	82.0	5.54	75.2	5.68	71.8	5.94	70.2	6.17	66.6
A-RAG	Trajectory	2.35	50.7	2.66	45.1	2.72	42.8	2.97	38.2	3.16	36.1
GRIP	Trajectory	3.79	55.0	3.99	50.7	4.16	49.6	4.21	47.1	4.41	45.6
UMA	Trajectory	16.22	98.6	19.59	**90.3**	20.35	83.6	20.83	76.3	20.94	69.9
Vanilla (BM25)	5	5.00	64.4	5.00	57.6	5.00	54.0	5.00	51.6	5.00	48.4
Vanilla (BM25)	10	9.99	86.9	10.00	74.0	10.00	69.2	10.00	64.9	10.00	60.6
Vanilla (BM25)	15	14.59	95.6	15.00	84.6	15.00	77.8	15.00	72.1	15.00	66.4
Vanilla (BM25)	20	17.65	**99.2**	20.00	88.6	20.00	**83.9**	20.00	76.6	20.00	71.9
SAADI (BM25)	5	5.00	69.7	5.00	62.8	5.00	59.2	5.00	55.7	5.00	52.3
SAADI (BM25)	10	9.99	88.9	10.00	77.7	10.00	74.3	10.00	68.8	10.00	64.3
SAADI (BM25)	15	14.59	96.4	15.00	88.0	15.00	82.4	15.00	77.0	15.00	70.9
SAADI (BM25)	20	17.65	**99.3**	20.00	**91.9**	20.00	**87.3**	20.00	**82.1**	20.00	**74.8**
Vanilla (Contriever)	5	5.00	71.8	5.00	59.9	5.00	54.0	5.00	47.9	5.00	44.3
Vanilla (Contriever)	10	9.99	91.4	10.00	76.1	10.00	68.3	10.00	60.7	10.00	54.0
SAADI (Contriever)	5	5.00	77.7	5.00	68.8	5.00	60.9	5.00	56.3	5.00	53.6
SAADI (Contriever)	10	9.99	93.7	10.00	83.7	10.00	77.4	10.00	68.4	10.00	64.0
QASPER											
RAG-R1	Trajectory	5.69	42.5	7.22	22.9	7.64	20.0	7.87	16.0	8.15	14.9
Agentic-R	Trajectory	3.52	33.9	3.76	18.8	3.76	16.1	3.83	13.6	3.87	11.1
A-RAG	Trajectory	3.40	32.1	3.40	15.5	3.48	12.6	3.46	10.2	3.54	8.9
GRIP	Trajectory	4.25	26.0	4.00	12.7	3.91	10.3	3.95	9.8	4.01	8.0
UMA	Trajectory	13.70	63.6	14.87	31.9	14.79	27.3	14.96	22.3	14.91	19.1
Vanilla (BM25)	5	5.00	33.8	5.00	17.9	5.00	14.4	5.00	12.8	5.00	11.5
Vanilla (BM25)	10	9.99	53.7	10.00	27.6	10.00	22.1	10.00	18.5	10.00	15.7
Vanilla (BM25)	15	14.90	69.0	15.00	34.0	15.00	27.2	15.00	23.1	15.00	18.4
Vanilla (BM25)	20	19.66	**79.6**	20.00	38.2	20.00	32.0	20.00	25.5	20.00	21.5
SAADI (BM25)	5	5.00	36.9	5.00	19.6	5.00	17.0	5.00	15.0	5.00	12.2
SAADI (BM25)	10	9.99	56.6	10.00	32.1	10.00	26.1	10.00	22.6	10.00	18.4
SAADI (BM25)	15	14.90	71.6	15.00	**39.6**	15.00	**32.3**	15.00	**26.6**	15.00	**22.0**
SAADI (BM25)	20	19.66	**82.3**	20.00	**44.7**	20.00	**36.3**	20.00	**30.0**	20.00	**24.5**
Vanilla (Contriever)	5	5.00	36.6	5.00	18.5	5.00	15.2	5.00	13.2	5.00	10.1
Vanilla (Contriever)	10	9.99	59.1	10.00	29.5	10.00	23.3	10.00	19.2	10.00	14.8
SAADI (Contriever)	5	5.00	37.1	5.00	19.6	5.00	16.2	5.00	13.1	5.00	11.4
SAADI (Contriever)	10	9.99	60.6	10.00	31.8	10.00	25.8	10.00	21.5	10.00	16.9

summary channel on MuSiQue-32K, from 74.0 to 76.6, and the memory channel contributes most on LooGLE, from 95.4 to 96.3. One caveat applies to the fixed weights used throughout. Because the weights are selected once and reused, two auxiliary channels that each leave a chunk inside the depth can together push it out, and Appendix F.9 gives an inspected instance.

F.8 Error analysis

We inspected the queries whose outcome changes between Vanilla and Full SAADI under BM25 at HR@10. Table 28 reports how the query population divides into rescues, regressions, and the queries where both runs succeed or both fail on each dataset.

How the inspected cases were labelled. Each inspected case is a query whose target crosses the top-10 boundary between the two runs, upward for a rescue and downward for a regression. For each one we first identify which fused channel moved the score, comparing the text, memory, and summary contributions between the two runs. Where the memory channel is responsible, we then

Table 27: Two ablations of SAADI, both reported at HR@10 under BM25 and both ending at the same full system. Panel (a) varies what the extraction step sees at index time while query-time scoring is held fixed, where History is the hierarchical document history and Records are the active-entity records recalled before extraction. Panel (b) varies which auxiliary channels enter the fused ranking while construction is held fixed, so Text only is the plain BM25 ranking over chunk text. The best value in each row is highlighted, with ties highlighted together. Values are percentages.

(a) Alternative state constructions query-time scoring fixed

Dataset	Chunk only	+ History	+ Memory	Full
LooGLE	96.1	96.2	96.0	**96.5**
MuSiQue-32K	75.6	**78.6**	74.9	77.7
QASPER-64K	23.3	25.8	23.7	**26.1**
NovelHopQA	21.4	**22.0**	21.3	**22.0**

(b) Alternative query-time scoring state construction fixed

Dataset	Text only	+ Memory	+ Summary	Full
LooGLE	95.4	96.3	96.1	**96.5**
MuSiQue-32K	74.0	75.4	76.6	**77.7**
QASPER-64K	22.1	24.5	24.4	**26.1**
NovelHopQA	20.0	20.9	21.8	**22.0**

Table 28: Query-level outcomes for Vanilla against full SAADI under BM25, as percentages of the cutoff-specific eligible paired queries. A rescue is a query that Vanilla fails and SAADI retrieves, and a regression is the reverse. The four outcomes partition the eligible queries within each cutoff, so rows sum to one hundred up to rounding.

Dataset	HR@5				HR@10			
	Rescue	Regression	Both success	Both fail	Rescue	Regression	Both success	Both fail
LooGLE	2.3	1.5	90.0	6.2	1.8	0.8	94.7	2.8
MuSiQue	7.1	1.9	55.7	35.3	6.6	2.9	71.1	19.4
QASPER	4.4	1.8	12.6	81.2	5.8	1.7	20.3	72.2
NovelHopQA	2.2	1.0	15.9	80.9	2.8	0.8	19.3	77.2

read the record field that produced the arg max, whether name, alias, or description, together with the term that matched, so that the attribution names a concrete string rather than a channel in the abstract. A rescue in which a film title entered a memory record as an alias and lifted a chunk whose own text never carried that title is labelled differently from one in which a summary term matched the query surface without bearing on the answer. Category labels were not fixed in advance. We inspected the cases first and grouped the observed mechanisms afterwards, so the categories are descriptive of this sample. All 40 cases were labelled by one author, without blinding to which run produced each ranking, and we therefore report the counts as an account of how the auxiliary signals act rather than as a measurement of how often each mechanism occurs.

Table 29 summarizes the resulting taxonomy over 20 stratified regression cases and 20 stratified rescue cases. Because both samples were stratified by construction, the proportions describe the inspected cases rather than the prevalence of each mechanism in the full population. Two categories recur on both sides and are worth separating from the rest. Fusion compounding shows that fixed auxiliary weights can fail where each channel alone succeeds, and the metric-defined cases show that a change in HR@10 does not always correspond to a change in whether answer-bearing evidence is available.

F.9 Qualitative analysis of retrieval regressions

Pattern A, semantic distractor promotion. An auxiliary signal promotes passages that are related to part of the query but do not satisfy its complete information need. MuSiQue query 1265 asks when the team associated with Trevor Dark last defeated the winner of the 1894–95 FA Cup. Vanilla ranks the answer-bearing chunk at 10 and adding Memory improves it to 8. The Summary signal instead promotes passages associated with the FA Cup portion of the query without preserving the complete Trevor Dark $\rightarrow$ team $\rightarrow$ opponent chain, and the answer-bearing chunk falls to rank 27 with Text+Summary and rank 24 with Full SAADI. The promoted passages match a salient query facet without matching the complete relational requirement.

Pattern B, summary granularity mismatch. The hierarchical summary is topically correct but too coarse to preserve the local detail the query needs. For QASPER query 923, "What is the

Table 29: Qualitative taxonomy of twenty stratified regression cases and twenty stratified rescue cases under BM25 at HR@10. A regression is a query that Vanilla retrieves and full SAADI does not, and a rescue is the reverse. Percentages describe the inspected samples rather than prevalence in the full query populations.

(a) Regression cases

Pattern	Description	n (%)
Semantic Distractor Promotion (A)	Auxiliary signals promote query-related passages that do not satisfy the full information need.	6 (30)
Summary Granularity Mismatch (B)	The summary preserves the broad topic but misses the fine-grained answer detail.	3 (15)
Entity/Event Ambiguity (C)	Memory identifies a related entity but not the requested occurrence, event, or relation.	4 (20)
Fusion Compounding (D)	Each auxiliary arm succeeds, but their combination moves the answer-bearing chunk outside the cutoff.	3 (15)
Metric-defined Regression (E)	HR@10 drops although useful answer-bearing evidence remains available or improves.	4 (20)

(b) Rescue cases

Pattern	Description	n (%)
Cross-hop Relation Bridging (A)	Auxiliary signals expose intermediate relations needed by multi-hop queries.	4 (20)
Semantic/Event Context Bridging (B)	Auxiliary context makes the relevant meaning or event more explicit than text matching alone.	6 (30)
Reference/Alias Resolution (C)	Aliases, roles, or terminology variants are linked to the same underlying entity or concept.	3 (15)
Summary Context Compression (D)	The summary compresses dispersed context into a representation that better matches the query.	5 (25)
Complementary Multi-signal Coverage (E)	Memory and Summary provide complementary evidence that jointly moves the answer-bearing chunk across the cutoff.	1 (5)
Metric-defined Rescue (F)	The answer-bearing chunk enters the top ten although evidence was already accessible under Vanilla.	1 (5)

architecture of the encoder?", the answer-bearing chunk states that the encoder contains M encoder blocks and that each block contains self-attention and a two-layer feed-forward network. Memory stays aligned with this and Text+Memory keeps the chunk within the top 10. The linked summary describes the model at a broader level as a Transformer-based sequence-to-sequence system, and Text+Summary moves the chunk from rank 8 to 16.

Pattern C, entity and event ambiguity. Memory identifies an appropriate entity, but entity-level relevance does not distinguish the exact occurrence the query asks about. For NovelHopQA query hop_1:566, the question asks what Tess requested Alec to do about the horse's pace. Memory correctly identifies Alec d'Urberville and holds descriptions linking him to Tess and their ride, yet Alec appears in many passages and events, and these matches do not localize the chunk in which Tess asks him to slow the horse to a walk. The answer-bearing chunk moves from rank 7 under Vanilla to rank 11 with Text+Memory and rank 16 with Full SAADI.

Pattern D, fusion compounding. Neither auxiliary signal individually causes a top-k failure and their combination does. In LooGLE query 1746 the answer-bearing chunk is ranked 8 by Vanilla, 10 by Text+Memory, and 9 by Text+Summary, so all three succeed under HR@10, while under Full SAADI it falls to rank 13. Several competing passages receive moderate advantages from both signals, and the combined contribution is enough to overtake an otherwise well-ranked chunk. These cases motivate query-adaptive or confidence-aware fusion in place of fixed auxiliary weights.

Pattern E, metric-defined regression. Not every HR@10 regression corresponds to a loss of useful evidence. For QASPER query 239, "Where do the news texts come from?", HR@10 records a regression because the annotated evidence chunk moves from rank 8 to 11. A supporting passage stating that the KPTimes newswire articles were collected by crawling selected online news websites moves from rank 31 under Vanilla to rank 1 under Full SAADI, and its Memory representation links "news texts" to online news websites. The metric reports a regression while retrieval of answer-bearing evidence improves.

F.10 Qualitative analysis of rescue cases

Rescues were inspected under the same protocol, so the proportions below describe the inspected sample rather than the full rescue population. Within that sample the most frequent mechanisms are semantic and event-context bridging at 30%, summary-based context compression at 25%, and cross-hop relation bridging at 20%.

Table 30: Regression cases under BM25, queries that Vanilla retrieves and full SAADI does not, broken down by which partial configuration of the query-time fusion ablation also regresses. Neither partial means the regression appears only once both channels are fused. Percentages are taken within each dataset and cutoff, and the Overall row pools the cases across datasets rather than averaging the four rows.

Dataset	HR@5				HR@10			
	+Memory only	+Summary only	Both partial	Neither partial	+Memory only	+Summary only	Both partial	Neither partial
LooGLE	3.6	35.7	25.0	35.7	14.3	64.3	14.3	7.1
MuSiQue	17.6	17.6	23.5	41.2	26.9	38.5	3.8	30.8
QASPER	16.7	50.0	25.0	8.3	21.7	43.5	13.0	21.7
NovelHopQA	55.6	33.3	0.0	11.1	28.6	28.6	14.3	28.6
Overall (micro)	16.7	35.9	21.8	25.6	22.9	44.3	10.0	22.9

Table 31: Representative examples of the qualitative regression patterns found during manual inspection at HR@10. Each case gives the rank of the answer-bearing chunk under Vanilla and under full SAADI, together with its rank under the two partial fusion configurations. Complete annotations for all twenty inspected cases are in the supplementary repository.

Label	Representative Case	Observed Failure	Evidence
A	MuSiQue 1265	**Semantic distractor promotion.** Summary favors passages matching the FA-Cup facet of the query without preserving the complete Trevor Dark → team → opponent reasoning chain.	Answer chunk: 10 → 24; T+M: 8; T+S: 27.
B	QASPER 923	**Summary granularity mismatch.** The answer-bearing chunk contains the requested encoder architecture, while its summary preserves only the broader Transformer-based model description.	Answer chunk: 8 → 16; T+M: 9; T+S: 16.
C	NovelHopQA `hop_1:566`	**Entity/event ambiguity.** Memory correctly identifies Alec d'Urberville and the horse-ride context, but does not uniquely localize the event where Tess asks him to slow the horse to a walk.	Answer chunk: 7 → 16; T+M: 11; T+S: 12.
D	LooGLE 1746	**Fusion compounding.** Both auxiliary configurations remain successful independently, but their combination promotes competing passages enough to move the answer-bearing chunk outside the cutoff.	Answer chunk: 8 → 13; T+M: 10; T+S: 9.
E	QASPER 239	**Metric-defined regression.** HR@10 decreases, although an answer-bearing supporting passage that directly identifies the source of the news texts is substantially improved by SAADI.	Answer chunk: 8 → 11; answer-bearing support: 31 → 1.

Cross-hop relation bridging. MuSiQue query 1321 requires traversing from the song *Better Than Me* to its performer and then to the relevant record label. The Memory representation associates Terry Dexter with performing the song, exposing an intermediate relation the complete information need requires.

Semantic and event context bridging. In LooGLE query 12 the answer-bearing chunk discusses Iqbal's criticism of nationalism in broad prose. Memory descriptions represent his condemnation of modern nationalism for dividing humanity and excluding religion from society, and the linked summary captures his rejection of nationalism and Western secularism, so the auxiliary representations align more directly with the intent of the query than the chunk text does.

Reference and alias resolution. QASPER query 184 uses the expanded phrase *out-of-vocabulary words* while the relevant section alternates between *OOV words* and *unknown words*. Memory links these expressions and bridges the terminology variants.

Table 32: Rescue cases under BM25, queries that full SAADI retrieves and Vanilla does not, broken down by which partial configuration of the query-time fusion ablation already rescues them. Neither partial means the rescue appears only once both channels are fused. Percentages are taken within each dataset and cutoff, and the Overall row pools the cases across datasets rather than averaging the four rows.

Dataset	HR@5				HR@10			
	+Memory only	+Summary only	Both partial	Neither partial	+Memory only	+Summary only	Both partial	Neither partial
LooGLE	38.1	14.3	33.3	14.3	33.3	9.1	48.5	9.1
MuSiQue	34.4	15.6	34.4	15.6	23.7	25.4	40.7	10.2
QASPER	36.2	19.0	27.6	17.2	39.0	20.8	26.0	14.3
NovelHopQA	5.0	55.0	30.0	10.0	12.0	32.0	44.0	12.0
Overall (micro)	32.6	20.7	31.5	15.2	29.9	21.6	36.6	11.9

Summary context compression. In QASPER query 1013 the answer sits inside a broader technical discussion of FinBERT pretraining data. The linked summary condenses the section into a statement that FinBERT was pretrained on Finnish news, discussion, and web data, producing a representation that aligns closely with the query.

Complementary multi-signal coverage. In QASPER query 1349 both auxiliary arms improve the answer-bearing chunk and neither individually reaches the top 10. The Summary representation captures several post-processing operations while Memory encodes related concepts such as lowercase settings and recasing, and Full SAADI moves the chunk from rank 36 to rank 10. We read this as complementary alignment between the auxiliary representations rather than evidence that either component independently caused the rescue.

Metric-defined rescue. In LooGLE query 1825 Full SAADI promotes the annotated evidence chunk from rank 11 to rank 6, while a separate passage directly describing the investigation is already ranked first by Vanilla. An HR@10 rescue can therefore reflect movement of the annotated evidence set without changing whether answer-bearing evidence reaches the downstream system.

Interpretation. These labels describe patterns observed during manual inspection and are not causal interventions on individual Memory entities, Summary representations, or fusion channels. Memory scoring aggregates separate name, alias, and description channels, and different linked items may contribute to different channels.

F.11 Notes on baseline treatment

Late Chunking. Late Chunking depends on a long-context embedding model, so it is run under `jina-embeddings-v3` only and appears in the Jina block of the grid in Appendix F.1 rather than under every backend. It is run at the three overlaps given in Appendix E, which appear as separate rows of that block, and the mean ranks of Appendix F.1 use the best of the three in each block rather than a single setting held fixed, so its ranked cells come from different configurations. This favours the baseline, and it never places Late Chunking above SAADI in any cell.

RAPTOR. RAPTOR retrieves summary nodes alongside passages, and comparing it at a fixed depth requires a conversion. We adopt the one most favourable to the method, charging a retrieved unit the number of chunks it covers while crediting it with all of them, and admitting an overflowing unit when it covers a target chunk. The resulting rows are a ceiling on what RAPTOR could reach at a given depth rather than an operating point, and they are read as such throughout.

QASPER conditions. The main QASPER rows are reported at the 64K expansion target, while the expansion tables report every target on one population, so QASPER values in the two places are read against different conditions rather than compared directly.

Table 33: Representative examples of the qualitative rescue mechanisms found during manual inspection at HR@10. Each case gives the rank of the answer-bearing chunk under Vanilla and under full SAADI, together with its rank under the two partial fusion configurations. Labels describe observed ranking behaviour and semantic alignment rather than causal effects of individual SAADI components.

Label	Representative Case	Observed Mechanism	Ranking Evidence
A	**MuSiQue-1321** *Who is the owner of the record label that the performer of Better Than Me is signed with?*	Memory makes the intermediate relation *Better Than Me* → Terry Dexter explicit, helping expose an intermediate hop required by the query.	Answer chunk: 11 → 1 T+M: 1; T+S: 9
B	**LooGLE-12** *What did Iqbal criticize about nationalism?*	Memory explicitly represents Iqbal's rejection of modern nationalism, while the linked summary captures the same broader semantic context, helping compensate for weaker surface lexical matching.	Answer chunk: 18 → 9 T+M: 12; T+S: 12
C	**QASPER-184** *What are the issues identified for out-of-vocabulary words?*	Memory connects *out-of-vocabulary words*, *OOV words*, and *unknown words*, bridging terminology differences between the query and relevant passages.	Answer chunk: 36 → 5 T+M: 3; T+S: 31
D	**QASPER-1013** *What was the new Finnish model trained on?*	The linked summary compresses the relevant section into a direct description of FinBERT being pretrained on Finnish news, online discussion, and web data.	Answer chunk: 12 → 7 T+M: 12; T+S: 7
E	**QASPER-1349** *What are the post-processing approaches applied to the output?*	Summary captures several post-processing operations, while Memory aligns with related concepts such as lower-case processing and recasing. Neither auxiliary arm alone reaches the cutoff, whereas Full SAADI does.	Answer chunk: 36 → 10 T+M: 23; T+S: 15
F	**LooGLE-1825** *Was there any progress in the investigation of Calvo Sotelo's murder?*	SAADI promotes the answer-bearing chunk, although a separate passage directly describing the investigation is already ranked first by Vanilla. The measured rescue therefore does not correspond to newly available answer-bearing evidence.	Answer chunk: 11 → 6 Answer-bearing evidence: Vanilla rank 1

Table 34: Retrieval success under BM25 for Vanilla, the two partial fusion configurations and full SAADI, as percentages of the cutoff-specific eligible paired queries. This extends the query-time fusion panel of Table 27 to HR@5, and the HR@10 columns repeat it. The best value in each row is highlighted.

Dataset	HR@5				HR@10			
	Vanilla	+Memory	+Summary	Full SAADI	Vanilla	+Memory	+Summary	Full SAADI
LooGLE	91.5	92.1	92.1	**92.3**	95.4	96.3	96.1	**96.5**
MuSiQue	57.6	61.2	59.6	**62.8**	74.0	75.4	76.6	**77.7**
QASPER	14.4	16.0	16.1	**17.0**	22.1	24.5	24.4	**26.1**
NovelHopQA	16.9	17.4	17.6	**18.1**	20.0	20.9	21.8	**22.0**