CONTINUAL GRAPH RETRIEVAL-AUGMENTED GENERATION UNDER CORPUS EXPANSION VIA PATH MEMORY

Anonymous authors
Paper under double-blind review

ABSTRACT

GraphRAG supports multi-hop reasoning for complex tasks, but becomes less reliable as both documents and queries evolve over time. Corpus expansion can weaken previously useful evidence paths while introducing new evidence that remains undiscovered, and evolving queries further shift graph exploration. These dynamics challenge both *retention* and *uptake* of supporting evidence, characterized by *evidence path dilution*. We propose EviWeaver, a non-parametric continual framework that maintains and refreshes *path memory* from prior graph exploration. EviWeaver preserves useful evidence paths and incorporates newly relevant support entirely at retrieval time without retraining. Experiments across three multi-hop question answering benchmarks show that EviWeaver improves retrieval robustness and answer quality under continual corpus and query evolution.

1 INTRODUCTION

In modern agentic systems, graph-based retrieval-augmented generation (GraphRAG) is increasingly important for connecting supporting evidence distributed across multiple documents through intermediate entities and facts (Li et al., 2025a; Edge et al., 2025; Guo et al., 2025; Gutiérrez et al., 2024). In practical scenarios, both the corpus and the queries evolve over time (Liska et al., 2022; Chen et al., 2023); new documents continuously expand the corpus and introduce novel nodes and edges into the graph, while new queries introduce changing information needs over the evolving knowledge base. Together, these continual changes reshape the retrieval landscape for GraphRAG.

Prior work has largely addressed continual GraphRAG reactively by updating the retrieval infrastructure, through incremental updates of graph structures (Zhu et al., 2025; Zhang et al., 2025) or retrievers (Gutiérrez et al., 2025; Zhang et al., 2026; Saravanan et al., 2026). However, such updates do not fully exploit knowledge accumulated through continual graph exploration. As the corpus and queries evolve, previously salient supporting evidence can become harder to recover, while newly introduced evidence may fail to be retrieved together with existing support. As illustrated in Figure 1, these effects can cause previously successful exploration traces to be overlooked and newly answerable queries to remain unresolved. We characterize this challenge as ***evidence path dilution***, or simply *path dilution*. Accordingly, the retrieval performance of existing continual GraphRAG approaches (Gutiérrez et al., 2025) declines over time.

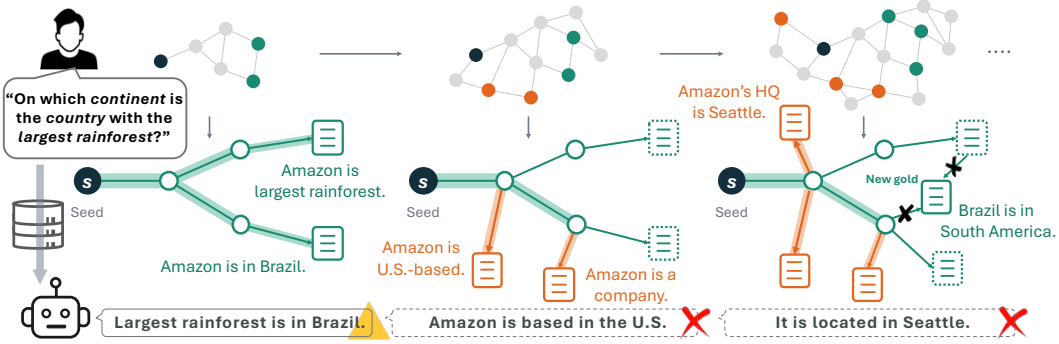


Figure 1: **Corpus expansion challenges both retention and uptake.** Amazon-related distractors displace prior evidence from the top-k, while new evidence on Brazil remains unretrieved.

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.

These observations suggest two key requirements for more practical continual GraphRAG: *retention* and *uptake*. The former maintains access to previously retrieved supporting evidence as the graph evolves, and the latter discovers newly available supporting evidence and integrates it with existing support. In this work, we introduce a novel, non-parametric framework **EviWeaver**, employing *path memory* to utilize useful evidence paths identified during continual retrieval. To capture informative paths beyond those directly exposed by the original query, EviWeaver broadens graph exploration using decomposed, query-relevant atomic contexts. It then applies backward tracing to identify edges that contributed to highly ranked retrieved documents and stores selected edges as path memory. As the graph evolves over time, EviWeaver reinforces and refreshes path memory to mitigate path dilution, updating retrieval seeds and stored edges while incorporating newly explored evidence paths. Most importantly, EviWeaver achieves this continual adaptation entirely at retrieval time, without retraining the retriever.

In brief, our contributions are summarized as follows.

- **Path dilution as a fundamental challenge in continual GraphRAG.** We characterize continual GraphRAG through *path dilution*, which captures how useful evidence paths become harder to recover or discover as the graph evolves. Building on the corresponding requirements of *retention* and *uptake*, we introduce query-expanding and delayed-evidence protocols that evaluate more realistic continual scenarios beyond existing simple fixed-query settings.
- **Path memory-based nonparametric framework.** Unlike prior approaches that mainly update graph structures or retrievers, we propose EviWeaver to exploit knowledge acquired during graph exploration through query-conditioned *path memory*. This allows EviWeaver to reinforce previously useful evidence paths against path dilution while incorporating newly explored paths as the graph evolves, entirely at retrieval time without retraining. Source code is available at `https://anonymous.4open.science/r/EVIWEAVER-A367`.
- **Robust performance and failure recovery.** On three multi-hop question answering benchmarks, EviWeaver significantly reduces performance degradation under corpus expansion by 60% in retrieval accuracy and by 53% in answer accuracy relative to the strongest baseline. Meanwhile, EviWeaver further pushes performance beyond the strongest baseline, improving retrieval by 2.1%, 1.7%, and 2.2%, and answer accuracy by 1.7%, 2.5%, and 2.3% under delayed-evidence, query-expanding, and fixed-query settings, respectively. We further analyze how evolving graph structure induces retrieval failures and how path memory helps recover missed supporting evidence.

2 Related Work

Graph Retrieval-Augmented Generation. Prior work improves GraphRAG through hierarchical or structured indexing (Sarthi et al., 2024; Edge et al., 2025; Li et al., 2025b), query-aware graph traversal and reasoning-path retrieval (Sun et al., 2024; Luo et al., 2024), and knowledge graph path selection for prompting (Liu et al., 2024). To handle growing corpora, recent methods integrate new documents into the graph incrementally (Guo et al., 2025; Luo et al., 2025; Zhu et al., 2025; Zhang et al., 2025) or propagate importance with personalized PageRank over an expandable graph (Gutiérrez et al., 2024; 2025). These methods update the graph as documents arrive but do not explicitly preserve the evidence paths of earlier queries, which path dilution weakens as the graph expands.

Continual Retrieval. Instead of continual pre-training or fine-tuning, which can incur high training cost and catastrophic forgetting (Shi et al., 2025), continual learning with external knowledge keeps new information outside model parameters (Wang et al., 2025; Zheng et al., 2025). Retrieval-based approaches follow this direction by updating retrievers and indices as the corpus evolves (Cai et al., 2023; Chen et al., 2023; Son et al., 2026; Zhang et al., 2026; Saravanan et al., 2026) or by deciding when retrievers should be updated (Ko et al., 2025), or incorporate temporal validity into retrieval (Nam et al., 2026). EviWeaver complements these approaches by maintaining query-conditioned path memory at retrieval time without retraining the retriever. For evaluation, HippoRAG2 fixes queries as the corpus grows (Gutiérrez et al., 2025), StreamingQA measures adaptation to new knowledge (Liska et al., 2022), and StreamBench evaluates LLMs on mixed-event document streams (Lee et al., 2026). Our protocols additionally let new queries arrive and delay part of their supporting evidence, so that retention and uptake are evaluated together.

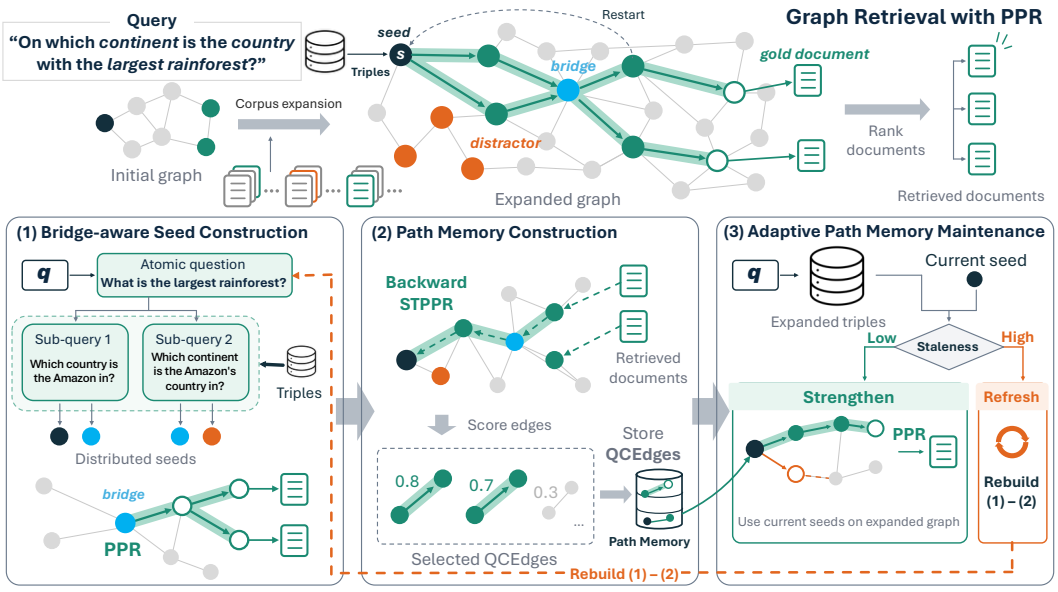


Figure 2: **An overview of the proposed framework.** EviWeaver constructs bridge-aware seeds, stores query-conditioned paths, and adaptively strengthens or refreshes them as the graph expands.

3 Problem Setting

We study retrieval robustness under *corpus expansion*, where new documents are continuously added and supporting evidence may be available when a query is first processed or arrive later. At each step t, a document batch $C^{(t)}$ arrives, yielding the cumulative indexed corpus $\bar{C}^{(t)} = \bigcup_{\tau=1}^{t} C^{(\tau)}$. The system incrementally maintains a triple-based knowledge graph $G^{(t)} = (V^{(t)}, E^{(t)})$ over $\bar{C}^{(t)}$, where nodes (e.g., entity and document) are connected by edges (e.g., relation, synonym, and source document). For a query q, let $t_0(q)$ denote the first step at which q is processed. At each step $t \geq t_0(q)$, the retriever returns a top-k document set $\mathcal{D}_q^{(t)}$ from the current graph $G^{(t)}$. We denote its retrieval quality by $R^{(t)}(q)$, measured by Recall@k over the query's supporting documents. Let $\mathcal{Q}$ denote the set of queries and T the final step. We evaluate each query at every step from its first appearance through step T. Let $\Omega = \{(q,t) \mid q \in \mathcal{Q}, t_0(q) \leq t \leq T\}$ collect these evaluations, where each (q,t) denotes evaluating query q at step t. Our goal is to maintain high retrieval quality throughout corpus expansion, preserving access to previously retrieved evidence (*retention*) while incorporating newly available evidence (*uptake*). We measure overall retrieval quality by averaging Recall@k over these evaluations, $\frac{1}{|\Omega|} \sum_{(q,t)\in\Omega} R^{(t)}(q)$.

4 Methodology

EviWeaver is a non-parametric framework built on a graph retriever that starts from query-related nodes and propagates importance over weighted edges to rank documents (Section 4.1). It adapts both the retrieval starting points and the edges emphasized during propagation. When a query first arrives, EviWeaver constructs diverse starting points to reach bridge entities (Section 4.2) and stores edges that contribute to highly ranked documents as query-specific path memory (Section 4.3). As new documents are added, EviWeaver either strengthens the stored edges to preserve access to previously retrieved evidence or refreshes path memory to incorporate newly available evidence (Section 4.4). Figure 2 illustrates the underlying retrieval process (top) and the three components of EviWeaver (bottom), using the query from Figure 1 as a running example.

4.1 Preliminaries: Graph Retrieval

We describe graph retrieval based on Personalized PageRank (PPR) (Page et al., 1998) on $G^{(t)}$ and how corpus expansion can change its starting points and propagation.

Starting points. Let $\mathcal{F}^{(t)}$ denote the cumulative triple store constructed from $\bar{\mathcal{C}}^{(t)}$, where each triple $f = (s, r, o)$ represents a relation r between a subject entity s and an object entity o. For a query q, triples are retrieved from $\mathcal{F}^{(t)}$ by cosine similarity between the query and triple embeddings, and each similarity score is assigned to the subject and object entity nodes of its triple. Normalizing these node scores yields the seed distribution $\mathbf{z}_q^{(t)}$ over $V^{(t)}$ to determine where graph retrieval starts.

Propagation and ranking. PPR propagates *importance*, the probability mass assigned to a node, from these starting points. Let $w^{(t)}(u, v)$ be the edge weight between nodes u and v, which is zero when the edge is absent. At each iteration, a fraction $\alpha \in (0, 1)$ of the total importance returns to the seed distribution and the remainder moves along graph transitions, whose probabilities are these edge weights normalized at each source node. The transition matrix $\mathbf{P}^{(t)}$ and the importance vector $\boldsymbol{\pi}_q^{(t)}$ therefore satisfy

$$P^{(t)}(u, v) = \frac{w^{(t)}(u, v)}{\sum_{v' \in V^{(t)}} w^{(t)}(u, v')}, \qquad \boldsymbol{\pi}_q^{(t)} = \alpha \mathbf{z}_q^{(t)} + (1 - \alpha)\big(\mathbf{P}^{(t)}\big)^\top \boldsymbol{\pi}_q^{(t)}. \tag{1}$$

Ranking document nodes by $\pi_q^{(t)}(d)$ gives the retrieved set $\mathcal{D}_q^{(t)}$. We call a sequence of transitions that carries importance from a seed node to a supporting document an *evidence path*.

Retrieval under corpus expansion. As new documents arrive, both $G^{(t)}$ and $\mathcal{F}^{(t)}$ expand. Newly added triples can change which triples the same query retrieves, and therefore its seed distribution $\mathbf{z}_q^{(t)}$. New edges also give each source node more outgoing transitions, so each existing transition carries a smaller share of importance, which we refer to as the path dilution problem. Together with the change in starting points, path dilution can lower the rank of a previously retrieved document even when its evidence path remains intact.

4.2 Bridge-Aware Seed Construction

Path memory identifies transitions that contribute to retrieving useful evidence. It is therefore important to give retrieval diverse starting points, so that it can explore paths that would otherwise be difficult to reach. These paths can lead to *bridge documents*, supporting documents that are reachable only through bridge entities not explicitly mentioned in the query. However, constructing seeds directly from the full query tends to concentrate seed weight on explicitly mentioned entities, because a query containing multiple constraints does not align well with an (s, r, o) triple that expresses a single relation. We therefore decompose the query into sub-queries aligned with individual triples. This broadens the starting points for retrieval, helping it reach such evidence and allowing path memory to record the transitions that lead to it.

Triple-aligned query decomposition. An LLM first extracts an *atomic question*, the earliest subproblem to resolve, expressed as a short question answerable by a single triple. The LLM also briefly describes the role of this question in the reasoning process and the subproblems that remain afterwards. Given the triples retrieved for the atomic question from $\mathcal{F}^{(t)}$, the LLM replaces complex expressions in the original query with concrete entities from these triples. Guided by this description, it then constructs sub-queries, each asking about a single relation. This makes entities the query does not mention available for retrieval, and expresses the query's constraints in a form that aligns with individual triples.

Seed construction and retrieval. We retrieve triples independently for each sub-query and sum the similarity scores of any triple retrieved by more than one sub-query. Assigning the aggregated scores to the subject and object entity nodes of each triple and normalizing the resulting node scores yields the seed distribution $\mathbf{z}_q^{(t)}$, which now spans the entities retrieved for different sub-queries. We run PPR on $G^{(t)}$ from this distribution to retrieve the top-k documents $\mathcal{D}_q^{(t)}$. This procedure runs whenever path memory is constructed, both when a query first arrives and when its memory is refreshed on the current graph.

4.3 PATH MEMORY CONSTRUCTION

Path dilution reduces the share of importance carried by an existing transition. To maintain the contribution of useful evidence paths, we store selected transitions at the edge level and increase their weights during later retrieval. This requires evaluating the transitions leading to retrieved documents, beyond the node importance provided by forward PPR. We therefore trace backward from the documents retrieved in Section 4.2, score the transitions leading to them, and store the selected ones as path memory.

Backward tracing. We apply backward tracing based on Single-Target PPR (STPPR) (Wang et al., 2020) to each document in $\mathcal{D}_q^{(t)}$. For each target document d, a *residue* at every node tracks the contribution to reaching d that remains to be propagated backward. We initialize the residue at d to one and all other residues to zero. Tracing proceeds against the direction of forward propagation, so a backward step from v to a neighboring node u corresponds to the forward transition $u \to v$. We record the residue transferred in this step as the flow contribution of $u \to v$,

$$\Delta\text{residue}(u) = \Delta\text{flow}(u \to v \mid d) = \frac{(1-\alpha_{\text{bw}})\rho_v}{\deg_t(u)}, \tag{2}$$

where ρ_v is the residue at v before propagation, α_{bw} is the STPPR restart probability, and $\deg_t(u)$ is the degree of u in the current graph. Accumulating these contributions over backward steps gives $\text{flow}(u \to v \mid d)$, which measures how strongly the transition $u \to v$ contributes to reaching d under backward tracing.

Query-conditioned edge(QCEdge) scores. Backward tracing distinguishes the transitions leading to each retrieved document, while the forward importance $\pi_q^{(t)}(d)$ determines how much that document contributes to the score. We combine the two over the retrieved documents to obtain

$$I(u, v \mid q) = \sum_{d \in \mathcal{D}_q^{(t)}} \pi_q^{(t)}(d)\, \text{flow}(u \to v \mid d). \tag{3}$$

A transition receives a high score when it contributes strongly to a document that also receives high importance under forward retrieval.

Preference for low-degree entities. At the first backward step from a document d, every adjacent transition shares the same residue $\rho_d = 1$ and the same document importance $\pi_q^{(t)}(d)$. Scores at this step are therefore inversely proportional to $\deg_t(u)$, favoring transitions through entities with few neighbors. Such an entity has few competing transitions, so it passes on a larger share of the importance it receives toward the evidence. Raising the weight of these transitions on an expanded graph helps limit the dispersion of importance into the neighbors added by expansion. Section 5.5 analyzes where the stored edges actually sit in the graph.

Path memory. Because score magnitudes vary across queries, each score is divided by the largest score for its query, giving the edge importance $\hat{I}(u, v \mid q)$. We store the top-B edges ranked by $I(u, v \mid q)$, together with their normalized importance, as the query-specific path memory $\mathcal{M}_q$, and call each stored edge a *QCEdge*. During later retrieval, EviWeaver raises the weights of these edges according to $\hat{I}(u, v \mid q)$. This selective strengthening helps importance reach the retrieved documents or stay near them, which can preserve access to existing evidence after the graph expands.

4.4 ADAPTIVE PATH MEMORY MAINTENANCE

EviWeaver groups the path memories of queries that first arrive at the same step into a *memory bank* m, and let t_m denote the step at which a bank m was built or last refreshed. As the corpus grows beyond t_m, the stored memory faces both changes described in Section 4.1. Shifts toward newly added triples may reveal useful connections not captured in memory, while additional competing transitions can reduce the share of importance carried by stored edges.

Memory staleness. To choose between strengthening and refresh, we use changes in triple retrieval as a signal of memory staleness. For the original query q, let $\mathcal{T}_q^{(t)}$ be its K most similar triples in $\mathcal{F}^{(t)}$ and $\sigma(q, f)$ the similarity between q and triple f, clipped at zero. We define s_{new} as the share of similarity mass assigned to triples absent from $\mathcal{F}^{(t_m)}$, and aggregate it over the bank's queries $\mathcal{Q}_M$ using the root mean square (RMS).

$$s_{\text{new}}(q \mid t_m) = \frac{\sum_{f \in \mathcal{T}_q^{(t)} \setminus \mathcal{F}^{(t_m)}} \sigma(q, f)}{\sum_{f \in \mathcal{T}_q^{(t)}} \sigma(q, f)}, \qquad s_{\text{mem}}(t) = \sqrt{\frac{1}{|\mathcal{Q}_M|} \sum_{q \in \mathcal{Q}_M} s_{\text{new}}(q \mid t_m)^2}. \quad (4)$$

Compared with the mean, RMS gives more weight to shifts concentrated in a subset of queries, while reacting less than the maximum to an isolated query (Appendix B.3). We strengthen the stored edges when $s_{\text{mem}}(t) < \tau_c$ and otherwise refresh the bank, where τ_c is the refresh threshold.

Strengthening. Each QCEdge is strengthened according to its normalized importance $\hat{I}(u, v \mid q)$, with a strengthening coefficient θ_t that adapts to degree growth around the stored edges. Since new neighbors at either endpoint add competing transitions in the undirected graph, we measure degree growth over the set V_M of distinct endpoints of all QCEdges in the bank. We set $\theta_t = g(t)^p$, where $p > 0$ controls how strongly strengthening responds to this growth.

$$\tilde{w}^{(t)}(u, v \mid q) = w^{(t)}(u, v) \exp\left(\theta_t \, \hat{I}(u, v \mid q)\right), \qquad g(t) = \frac{\sum_{v \in V_M} \deg_t(v)}{\sum_{v \in V_M} \deg_{t_m}(v)}. \quad (5)$$

Here $w^{(t)}(u, v)$ is the edge weight before strengthening. EviWeaver then reconstructs the seed distribution from the original query over $\mathcal{F}^{(t)}$, without repeating query decomposition, and runs PPR with the modified weights applied only to retrieval for q.

Refresh. When staleness reaches the threshold, EviWeaver reruns bridge-aware seed construction (Section 4.2) and path memory construction (Section 4.3) on the unmodified graph $G^{(t)}$ for every query in the bank. For each query, it returns the retrieved documents and replaces the stored path memory with the rebuilt one. Finally, t_m is updated to t so that later steps measure staleness and degree growth against the new memory.

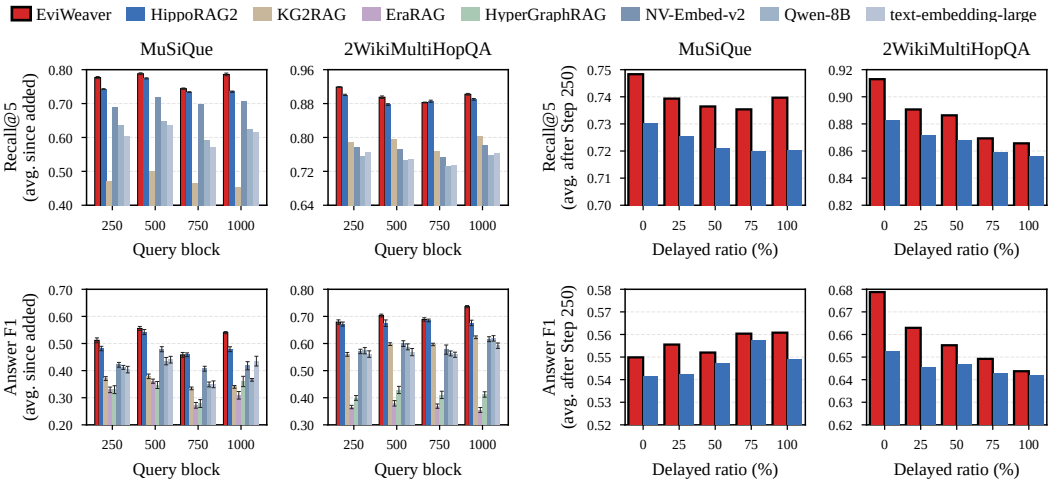
5 Experiments

5.1 Setup

Datasets and Expanding Corpus. We use three popular benchmark datasets, MuSiQue (Trivedi et al., 2022), 2WikiMultiHopQA (Ho et al., 2020), and HotpotQA (Yang et al., 2018), to build the corpus from their supporting documents and distractors. Following Gutiérrez et al. (2025), for each dataset, 1000 questions are split into four blocks, of which documents are added to the graph at each expansion step, indexed by the cumulative number of questions $t \in \{250, 500, 750, 1000\}$.

Continual Evaluation Protocols. The existing *fixed-query* setting evaluates the first query block at every step as the corpus expands (Gutiérrez et al., 2025). Documents added after the first step are not supporting documents for these queries, so this setting isolates retention. It assumes that the query set does not change and that all supporting documents are present from the first step. We introduce two settings that relax these assumptions. The ***query-expanding*** setting adds a new query block with its documents at each step and evaluates every block from its introduction step onward, measuring both performance on new queries and retention on earlier ones. In the ***delayed-evidence*** setting, about half of the supporting documents of each delayed query are available at the first step and the rest arrive over later steps. The delayed documents are chosen at random regardless of their position in the reasoning chain, so even the first-hop document can arrive late. This matches practical corpus growth, in which documents need not arrive in the order of a query's reasoning chain. We vary the delayed-query ratio, the fraction of queries whose documents are delayed, over $\{0, 25, 50, 75, 100\}\%$.

Baselines and Metrics. We compare EviWeaver with the dense retrievers text-embedding-3-large, Qwen3-Embedding-8B, and NV-Embed-v2 (Lee et al., 2025), and with the graph-based retrievers KG2RAG (Zhu et al., 2025) and HippoRAG2 (Gutiérrez et al., 2025). All methods receive the



(a) Query-expanding continual setting.

(b) Delayed-evidence continual setting.

Figure 3: **Retrieval and answer quality under query expansion and delayed evidence.** (a) Each query block is averaged over the steps from its introduction. (b) EviWeaver and HippoRAG2 are averaged over steps 500–1000.

same corpus at each step and provide their top-5 retrieved units to the same QA reader. We report Recall@5 over supporting documents and token-level answer F1. Since EraRAG (Zhang et al., 2025) and HyperGraphRAG (Luo et al., 2025) do not return document rankings, only their answer F1 is reported. In the delayed-evidence setting, Recall@5 is computed against all supporting documents, including those not yet released.

Implementation Details. GPT-4o-mini with temperature 0 is used for all LLM calls and NV-Embed-v2 is used for default dense retrieval. EviWeaver uses $\alpha = 0.5$, $B = 30$, $p = 2$, and $\tau_c = 0.35$ for all datasets and steps. Fixed-query and query-expanding results are averaged over five preprocessing runs. All other experiments use a single run shared across methods. See more details in Appendix A.

5.2 Main Results

Figure 3 reports results on MuSiQue and 2WikiMultiHopQA. On HotpotQA, HippoRAG2 and EviWeaver both reach near-saturated Recall@5 that changes little under corpus expansion (Appendix D).

Query-Expanding Setting. In Figure 3(a), EviWeaver achieves the highest Recall@5 and answer F1 on most query blocks. Averaged over all queries at each step after their introduction, EviWeaver reaches 0.775 Recall@5 on MuSiQue and 0.903 on 2WikiMultiHopQA, compared with 0.750 and 0.890 for HippoRAG2. EviWeaver stays ahead of HippoRAG2 in Recall@5 on every block at all steps after its introduction step, for both datasets (Appendix D.2), showing better retention of earlier queries as new blocks and their documents arrive.

Delayed-Evidence Setting. In Figure 3(b), EviWeaver achieves higher Recall@5 and answer F1 than HippoRAG2 at every delayed-query ratio, with Recall@5 gaps of 1.4–1.9 points on MuSiQue and 0.9–3.0 points on 2WikiMultiHopQA. On MuSiQue, EviWeaver leads at every step for every ratio. On 2WikiMultiHopQA, the gap narrows as more queries are delayed. When most queries are delayed, their path memory is first built with only half of their supporting documents, and EviWeaver falls briefly below HippoRAG2 at step 500. It regains the lead from step 750 onward, and the gap is largest at step 1000 (Appendix D.3). Path memory built on partial evidence thus does not prevent the uptake of later documents. On MuSiQue, answer F1 does not follow the trend of Recall@5 across ratios, since successful answering requires both complete retrieval of the two to four supporting documents and their proper ordering in the context.

Fixed-Query Setting. From step 250 to 1000, the Recall@5 of HippoRAG2 decreases by 7.4% on MuSiQue and 11.5% on 2WikiMultiHopQA, compared with 4.1% and 3.8% for EviWeaver (Figure 4). On 2WikiMultiHopQA, HippoRAG2 starts higher, but EviWeaver overtakes it from step

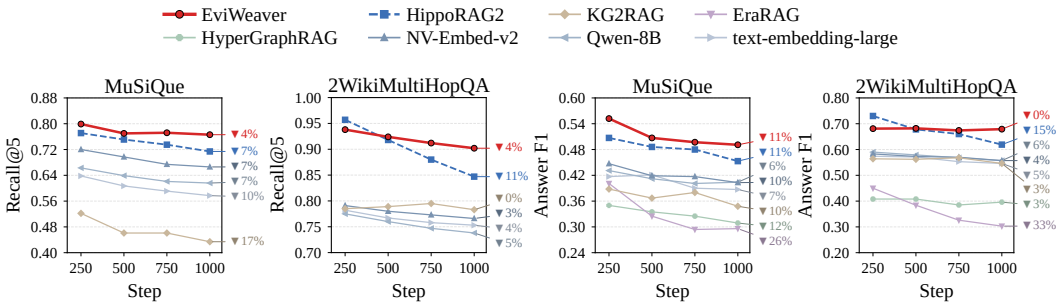


Figure 4: **Retrieval and answer quality in the fixed-query setting.** Numbers indicate the relative decrease from step 250 to step 1000.

Table 1: **Ablation analysis of EviWeaver on Recall@5.** Results are averaged over the fixed-query and delayed-evidence settings. Bold and underline mark the best and second-best post-250 results.

Dataset	Step	EviWeaver	w/o QCEdge	w/o QD	w/o Refresh	w/o Adaptive Strength
MuSiQue	250	0.667	0.667	0.654	0.667	0.667
	500	**0.736**	0.719	<u>0.728</u>	0.726	**0.736**
	750	**0.746**	0.733	0.732	0.738	<u>0.742</u>
	1000	**0.748**	0.724	0.731	0.738	0.740
2WikiMultiHopQA	250	0.690	0.690	0.698	0.690	0.690
	500	0.872	<u>0.874</u>	0.869	**0.875**	0.871
	750	**0.906**	0.884	<u>0.903</u>	0.901	0.902
	1000	**0.894**	0.851	<u>0.883</u>	0.877	0.881

500 and keeps answer F1 almost unchanged, whereas the F1 of HippoRAG2 decreases by 15.2%. On MuSiQue, EviWeaver achieves the highest F1 at every step, although its F1 decreases by a similar ratio to that of HippoRAG2.

5.3 Ablation Study

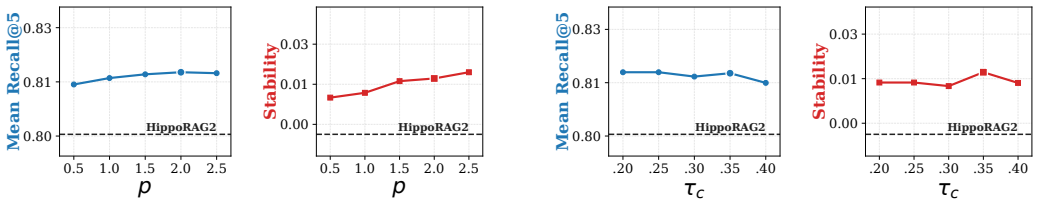
Table 1 compares EviWeaver with four variants. *w/o QCEdge* applies bridge-aware seed construction only at the construction step and stores no path memory, so it has nothing to strengthen or refresh afterwards. *w/o QD* constructs seeds and path memory from the original query without triple-aligned decomposition. *w/o Refresh* always strengthens the current memory, and *w/o Adaptive Strength* applies the strengthening coefficient of the first expansion step to all later steps.

Seed and Path Memory Construction. On MuSiQue and 2WikiMultiHopQA, removing path memory causes the largest drop at the final step, making it the main source of robustness to corpus expansion. Removing decomposition also lowers later-step performance. On 2WikiMultiHopQA, decomposition does not improve initial retrieval but still helps after expansion, consistent with its role in shaping which transitions path memory stores.

Adaptive Memory Maintenance. *w/o Adaptive Strength* falls behind EviWeaver at later steps, consistent with continued degree growth around stored edges. *w/o Refresh* also falls behind at later steps, indicating that memory built on an earlier graph becomes less effective even when strengthened.

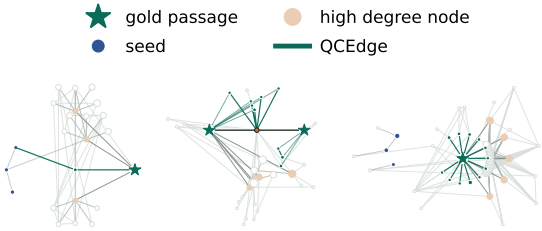
5.4 Parameter Sensitivity

We analyze the strengthening exponent p and refresh threshold τ_c. Step 250 is excluded because strengthening and refresh begin only after the first expansion. As shown in Figure 5, EviWeaver outperforms HippoRAG2 at every tested value of both parameters, and mean Recall@5 varies by at most 0.3 points across each range. Mean Recall@5 peaks at $p = 2$, while stability continues to increase with p. $\tau_c = 0.35$ achieves the highest stability with mean Recall@5 close to the best setting. Lower thresholds refresh more often and repeat query decomposition at each refresh. We therefore use $p = 2$ and $\tau_c = 0.35$ throughout, with additional sensitivity results in Appendix D.4.



(a) Strengthening exponent p. (b) Refresh threshold τ_c.

Figure 5: **Hyperparameter sensitivity.** Mean Recall@5 over steps 500–1000 and stability ($R_{1000} - R_{500}$), averaged over both datasets and all delayed-query ratios.



(a) Entrance. (b) Bridge. (c) Pool.

		Entry degree		Seed neighbors		Entity degree	
	Entries	250	1000	Orig.	Bridge	QCEdge	Other
Kept	2.39	20	93	—	—	—	—
Lost	1.63	48	242	—	—	—	—
Recovered	1.50	36	166	0.67	2.55	—	—
Unrecovered	1.66	50	252	0.52	1.04	—	—
Entrance	2.16	22	92	—	—	5	51
Bridge	1.00	38	158	—	—	3	44
Pool	1.29	45	183	—	—	5	31

(d) Lost and recovered passages.

Figure 6: **Failure-recovery case analysis.** (a)–(c) show one gold passage per case type at step 1000 with the QCEdges of its query. (d) shows mean entries and median degrees.

5.5 Where Does Path Memory Help?

Why are passages lost? As the corpus expands, HippoRAG2 drops some gold passages from its top-5 although they remain in the graph (Appendix D.6). On a shortest path from the seeds to a gold passage, we call the entity one hop before the passage an *entry entity*. Lost passages have fewer entry entities than kept ones, and these entities already have higher degree at the initial step (first block of Figure 6d). Path dilution thus first pushes out passages whose evidence paths enter through few entities that already divide importance among many transitions.

How are they recovered? EviWeaver returns some lost passages to the top-5, and these passages differ from unrecovered ones in two properties (second block). First, bridge-aware seed construction (Section 4.2) decomposes the query to widen the starting points and add bridge entities as seeds. For recovered passages, this places far more seeds right next to the passage. Second, recovered passages have lower-degree entry entities, in line with how QCEdges are built. Backward tracing (Section 4.3) traces importance back from the retrieved documents and stores the highest-scoring transitions as QCEdges, favoring low-degree entities. QCEdges thus touch entities of far lower degree than other nearby entities (third block). Recovery falls into three situations by where QCEdges lie (Figure 6a–c). In Entrance, QCEdges lead from the seeds into an entry entity, so strengthening acts on the evidence path itself. In Bridge, QCEdges surround an entity shared with other retrieved documents, so importance reaches the passage from those documents. In Pool, QCEdges lie among nearby entities instead of the high-degree entry entities. With few competing transitions, these edges keep importance near the passage as the graph grows. Bridge and Pool passages have about one entry entity, so recovery is not limited to passages with many entry entities.

6 Conclusion

We introduced EviWeaver, a non-parametric continual GraphRAG framework that exploits query-conditioned path memory through adaptive strengthening and refresh. Beyond the standard fixed-query evaluation protocol, we proposed more realistic settings featuring expanding queries and delayed evidence to assess the system's adaptation capability to previously unavailable information. Across three multi-hop question answering (MHQA) benchmarks, EviWeaver consistently mitigates performance degradation while maintaining strong retrieval and answer quality.

AI USE STATEMENT

We used large language models (LLMs) solely to assist with manuscript writing and revision, including improvements to clarity and fluency. All research ideas, methodological contributions, experimental designs, analyses, and reported results are original work by the authors.

ETHICS STATEMENT

Our work does not include experiments or analysis with human subjects or private data. We do not see any specific ethical concerns that can arise from the proposed framework.

REPRODUCIBILITY STATEMENT

Our study uses publicly available multi-hop QA benchmarks, including MuSiQue, 2WikiMultiHopQA, and HotpotQA. We use a low temperature to reduce variability in LLM outputs. We provide the full algorithm, experimental settings, and hyperparameters in Appendix A, and computing resources in Appendix E. LLM prompts are provided in Appendix F. The assumptions and complete proofs underlying our theoretical analysis are included in Appendix B.

REFERENCES

Yinqiong Cai, Keping Bi, Yixing Fan, Jiafeng Guo, Wei Chen, and Xueqi Cheng. L^2R: Lifelong learning for first-stage retrieval with backward-compatible representations. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023.

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 GraphRAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2025.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. LightRAG: Simple and fast retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. HippoRAG: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 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. In *International Conference on Machine Learning*, 2025.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2020.

Dayoon Ko, Jinyoung Kim, Sohyeon Kim, Jinhyuk Kim, Jaehoon Lee, Seonghak Song, Minyoung Lee, and Gunhee Kim. When should dense retrievers be updated in evolving corpora? Detecting out-of-distribution corpora using GradNormIR. In *Findings of the Association for Computational Linguistics: ACL 2025*, 2025.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. NV-Embed: Improved techniques for training LLMs as generalist embedding models. In *International Conference on Learning Representations*, 2025.

Yukyung Lee, Yebin Lim, Woojun Jung, Wonjun Choi, and Susik Yoon. Can structural cues save llms? evaluating language models in massive document streams. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, 2026.

Siyuan Li, Yongping Du, and Mingyang Li. Graph-augmented retrieval with memory-driven reasoning and constraint-aware filtering for MultiHop QA. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025a.

Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. StructRAG: Boosting knowledge intensive reasoning of LLMs via inference-time hybrid information structurization. In *International Conference on Learning Representations*, 2025b.

Adam Liska, Tomas Kocisky, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien De Masson D'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsenan-Mcmahon, Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. StreamingQA: A benchmark for adaptation to new knowledge over time in question answering models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162. PMLR, 2022.

Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. Knowledge graph-enhanced large language models via path selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Haoran Luo, Haihong E, Guanting Chen, Yandan Zheng, Xiaobao Wu, Yikai Guo, Qika Lin, Yu Feng, Zemin Kuang, Meina Song, Yifan Zhu, and Anh Tuan Luu. HyperGraphRAG: Retrieval-augmented generation via hypergraph-structured knowledge representation. In *Advances in Neural Information Processing Systems*, 2025.

Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *International Conference on Learning Representations*, 2024.

Youngeun Nam, Joeun Kim, Hwanjun Song, Susik Yoon, Jae-Gil Lee, and Byung Suk Lee. TimelyRAG: Semantic-temporal hybrid retrieval for time-critical question answering in overlapping-evolving documents. *arXiv preprint arXiv:2609.11572*, 2026.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bring order to the web. In *Proc. of the 7th International World Wide Web Conf*, 1998.

Manoj Saravanan, Rohit Kumar Salla, and Ramya Manasa Amancherla. AURORA: Neuro-symbolic continual indexing for evolving RAG systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

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

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, and Hao Wang. Continual learning of large language models: A comprehensive survey. *ACM Computing Surveys*, 58(5), 2025.

HuiJeong Son, Hyeongu Kang, Sunho Kim, Subeen Ho, SeongKu Kang, Dongha Lee, and Susik Yoon. CREAM: Continual retrieval on dynamic streaming corpora with adaptive soft memory. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, 2026.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M. Ni, Heung-Yeung Shum, and Jian Guo. Think-on-Graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, 2024.

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, 2022.

Hanzhi Wang, Zhewei Wei, Junhao Gan, Sibo Wang, and Zengfeng Huang. Personalized PageRank to a target node, revisited. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

Mingyang Wang, Alisa Stoll, Lukas Lange, Heike Adel, Hinrich Schuetze, and Jannik Strötgen. Bring your own knowledge: A survey of methods for LLM knowledge expansion. In *Proceedings of the First Workshop on Large Language Model Memorization (L2M2)*, 2025.

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

Fangyuan Zhang, Zhengjun Huang, Yingli Zhou, Qintian Guo, Zhixun Li, Wensheng Luo, Di Jiang, Yixiang Fang, and Xiaofang Zhou. EraRAG: Efficient and incremental retrieval augmented generation for growing corpora. *arXiv preprint arXiv:2506.20963*, 2025.

Senlei Zhang, Tongjun Shi, Dandan Song, Luan Zhang, Shuhao Zhang, Xiaofei Liao, and Hai Jin. FlowRAG: Continual learning for dynamic retriever in retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2026*, 2026.

Junhao Zheng, Shengjie Qiu, Chengming Shi, and Qianli Ma. Towards lifelong learning of large language models: A survey. *ACM Computing Surveys*, 57(8), 2025.

Xiangrong Zhu, Yuexiang Xie, Yi Liu, Yaliang Li, and Wei Hu. Knowledge graph-guided retrieval augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2025.

A Implementation Details

A.1 Graph, Seeds, and Path Memory

Indexing. We follow the indexing of HippoRAG2 (Gutiérrez et al., 2025). The graph is undirected and has three edge types. Relation edges link the subject and object entities of each triple extracted by OpenIE and are weighted by the number of such triples. Document edges link each document to the entities extracted from it with weight one. Synonym edges link entity pairs whose embedding similarity exceeds 0.8 and are weighted by that similarity. At each step, only the new batch is indexed. Its entities merge with existing nodes of the same string, and synonym edges are added between its new entities and all existing ones. Earlier nodes and edges are kept.

Seed construction. During bridge-aware seed construction, an LLM filters the candidate triples retrieved for each sub-query. The scores of the retained triples are summed across sub-queries and min–max normalized before being assigned to their subject and object entity nodes, and normalizing the resulting node scores gives $\mathbf{z}_q^{(t)}$. During strengthening, the same filtering and normalization are applied to the triples retrieved for the original query. If filtering retains no triple, retrieval falls back to dense retrieval and memory construction returns $\mathcal{M}_q = \emptyset$. Sub-queries are discarded after seed construction and regenerated whenever memory is refreshed.

Backward tracing. Backward tracing uses the unweighted degree so that edge scores reflect graph structure regardless of edge-weight scales. It stops after $L = \lceil \log \epsilon / \log(1 - \alpha_{\mathrm{bw}}) \rceil + 1$ levels, so that $(1 - \alpha_{\mathrm{bw}})^{L-1} \leq \epsilon$ for a tolerance ϵ. We use $\alpha_{\mathrm{bw}} = 0.5$ and $\epsilon = 10^{-4}$.

Memory maintenance. Each memory bank is maintained independently, so the query-expanding protocol maintains one bank per query block. Staleness uses triples ranked by dense similarity before LLM filtering, so computing $s_{\mathrm{mem}}(t)$ requires no LLM call.

Parameter settings. We use $B = 30$, $p = 2$, and $\tau_c = 0.35$, with $K = 5$ for memory staleness estimation, across all datasets and expansion steps.

A.2 Baselines

Table 2 lists the retrieval unit of each baseline and how it updates its index at each step. No method rebuilds its index from scratch, and other hyperparameters follow the official implementations.

Table 2: Retrieval unit and per-step index update of each baseline.

Method	Retrieval unit	Index update per step
Dense retrievers	Document	Append document embeddings
KG2RAG	Chunk	Append chunks and triples to the graph
HippoRAG2	Document	Add triples with their entity and document nodes
EraRAG	Summary or graph node	Insert into hashed segments and re-summarize them
HyperGraphRAG	Hyperedge fact	Append entities and hyperedges

A.3 Overall Procedure

Algorithm 1 shows the procedure for one memory bank. Initial construction and refresh share the same steps. BuildBridgeAwareSeed runs atomic question extraction, decomposition, and seed construction. BuildQuerySeed builds seeds from the original query without decomposition. Retrieve$(w, \mathbf{z})$ runs PPR with edge weights w from $\mathbf{z}$ and returns the importance vector and the top-k documents. ExtractPathMemory runs backward tracing and keeps the top-B edges.

Algorithm 1 Overall procedure of EviWeaver for one memory bank

Require: Bank $\mathcal{Q}_M$ arriving at t_0; $\{G^{(t)}, \mathcal{F}^{(t)}\}_{t=t_0}^{T}$; k, B, p, τ_c
Ensure: $\mathcal{D}_q^{(t)}$ for all $q \in \mathcal{Q}_M$ and $t_0 \le t \le T$

```
for t = t_0, ..., T do
    if t = t_0 or s_mem(t) ≥ τ_c then                                  ▷ Construct or refresh
        for all q ∈ Q_M do
            z_q^(t) ← BUILDBRIDGEAWARESEED(q, F^(t))
            (π_q^(t), D_q^(t)) ← RETRIEVE(w^(t), z_q^(t))
            M_q ← EXTRACTPATHMEMORY(G^(t), π_q^(t), D_q^(t))
        end for
        t_m ← t
    else                                                               ▷ Strengthen
        θ_t ← g(t)^p
        for all q ∈ Q_M do
            z_q^(t) ← BUILDQUERYSEED(q, F^(t))
            w̃^(t)(·, · | q) ← strengthen M_q by Eq. (5)
            (π_q^(t), D_q^(t)) ← RETRIEVE(w̃^(t)(·, · | q), z_q^(t))
        end for
    end if
end for
return {D_q^(t)}
```

B Supporting Analyses

This appendix formalizes path dilution (Section 4.1) and gives the rationale for three design choices of adaptive path memory maintenance (Section 4.4), namely the strengthening coefficient adapted to degree growth, exponential strengthening, and RMS aggregation for refresh. We write $S_t(u) = \sum_v w^{(t)}(u, v)$ for the total weight of the edges incident to u, so that $P^{(t)}(u, v) = w^{(t)}(u, v)/S_t(u)$ in Eq. (1).

B.1 Path Dilution and Degree Growth

Path contributions to PPR. Because $\mathbf{P}^{(t)}$ has row sums at most one and $0 < 1 - \alpha < 1$, Eq. (1) has the solution

$$\boldsymbol{\pi}_q^{(t)} = \alpha \sum_{\ell=0}^{\infty} (1-\alpha)^{\ell} \big((\mathbf{P}^{(t)})^{\top}\big)^{\ell} \mathbf{z}_q^{(t)}.$$

Let Γ_d denote the set of paths $\gamma = (v_0, \ldots, v_L)$ of any length $L \ge 0$ that end at d, where a path may revisit nodes. Expanding the matrix powers entrywise gives

$$\pi_q^{(t)}(d) = \sum_{\gamma \in \Gamma_d} c_t(\gamma), \qquad c_t(\gamma) = \alpha(1-\alpha)^L \, z_q^{(t)}(v_0) \prod_{i=0}^{L-1} P^{(t)}(v_i, v_{i+1}). \tag{6}$$

The score of a document is therefore the total contribution of the paths that reach it. Each contribution contains the product of the transition probabilities along the path.

Lemma 1 (Compounding transition dilution)**.** *Let $\gamma = (v_0, \ldots, v_L)$ be a path in $G^{(t_m)}$ whose edge weights are unchanged at step t, so that $w^{(t)}(v_i, v_{i+1}) = w^{(t_m)}(v_i, v_{i+1}) > 0$ for $i = 0, \ldots, L-1$. Let $\Delta_i = S_t(v_i) - S_{t_m}(v_i) \ge 0$ be the edge weight that expansion adds at v_i.*

(i) The transition probabilities along γ satisfy

$$\frac{\prod_{i=0}^{L-1} P^{(t)}(v_i, v_{i+1})}{\prod_{i=0}^{L-1} P^{(t_m)}(v_i, v_{i+1})} = \prod_{i=0}^{L-1} \frac{S_{t_m}(v_i)}{S_{t_m}(v_i) + \Delta_i} \le 1, \tag{7}$$

with strict inequality if and only if $\Delta_i > 0$ for some i.

(ii) If $z_q^{(t)}(v_0) = z_q^{(t_m)}(v_0)$, then $c_t(\gamma)$ in Eq. (6) equals $c_{t_m}(\gamma)$ multiplied by the same factor.

Proof. For each i, the unchanged edge weight cancels in

$$\frac{P^{(t)}(v_i, v_{i+1})}{P^{(t_m)}(v_i, v_{i+1})} = \frac{w^{(t_m)}(v_i, v_{i+1})/\big(S_{t_m}(v_i) + \Delta_i\big)}{w^{(t_m)}(v_i, v_{i+1})/S_{t_m}(v_i)} = \frac{S_{t_m}(v_i)}{S_{t_m}(v_i) + \Delta_i}.$$

This ratio lies in $(0, 1]$ and equals one if and only if $\Delta_i = 0$. Multiplying over i gives (i). In $c_t(\gamma)/c_{t_m}(\gamma)$, the factor $\alpha(1-\alpha)^L$ and the seed weight $z_q(v_0)$ cancel, which gives (ii). □

Interpretation. Lemma 1 formalizes path dilution. Even when the edge weights and the seed weight of a path are unchanged, edge weight added at its nodes reduces its contribution. This dilution compounds multiplicatively along the path. If the total edge weight at every node of γ grows by the same factor $c \geq 1$, the factor in Eq. (7) becomes c^{-L}. Paths with more transitions are therefore diluted more.

Degree growth. For a node v with $\deg_t(v) > 0$, let $\bar{w}_t(v) = S_t(v)/\deg_t(v)$ denote its mean incident-edge weight. Then

$$\frac{S_t(v)}{S_{t_m}(v)} = \frac{\deg_t(v)}{\deg_{t_m}(v)} \cdot \frac{\bar{w}_t(v)}{\bar{w}_{t_m}(v)}. \tag{8}$$

For the endpoint set V_M of a memory bank, define

$$\bar{W}_t = \frac{\sum_{v \in V_M} S_t(v)}{\sum_{v \in V_M} \deg_t(v)}.$$

For the degree growth $g(t)$ in Eq. (5),

$$\frac{\sum_{v \in V_M} S_t(v)}{\sum_{v \in V_M} S_{t_m}(v)} = g(t) \cdot \frac{\bar{W}_t}{\bar{W}_{t_m}}. \tag{9}$$

Both identities follow directly from $S_t(v) = \deg_t(v)\bar{w}_t(v)$. When the mean incident-edge weight at v_i is unchanged, the factor $S_{t_m}(v_i)/\big(S_{t_m}(v_i) + \Delta_i\big)$ in Lemma 1 therefore equals the inverse of the degree growth at v_i. We accordingly use $g(t)$ to measure the growth in competition around stored edges. The exponent p in $\theta_t = g(t)^p$ controls how strongly strengthening responds to this growth.

B.2 Exponential Strengthening

Strengthening multiplies $w^{(t)}(u, v)$ by $\exp(a_e)$ for each QCEdge $e = \{u, v\} \in \mathcal{M}_q$, where $a_e = \theta_t \hat{I}(u, v \mid q)$ (Eq. (5)). Let $\tilde{P}^{(t)}$ denote the transition probabilities computed from the strengthened weights $\tilde{w}^{(t)}(\cdot, \cdot \mid q)$.

Lemma 2 (Log-ratio shift under strengthening)**.** *Let $e = \{u, v\} \in \mathcal{M}_q$.*

(i) For every edge $\{u, z\} \notin \mathcal{M}_q$ with $w^{(t)}(u, z) > 0$,

$$\frac{\tilde{P}^{(t)}(u, v)}{\tilde{P}^{(t)}(u, z)} = \exp(a_e)\, \frac{P^{(t)}(u, v)}{P^{(t)}(u, z)}. \tag{10}$$

(ii) If e is the only QCEdge incident to u and $P = P^{(t)}(u, v)$ satisfies $0 < P < 1$, then

$$\tilde{P}^{(t)}(u, v) = \frac{P \exp(a_e)}{1 - P + P \exp(a_e)}, \qquad \log \frac{\tilde{P}^{(t)}(u, v)}{1 - \tilde{P}^{(t)}(u, v)} = \log \frac{P}{1 - P} + a_e. \tag{11}$$

Proof. Let $\tilde{S}_t(u)$ be the total edge weight at u after strengthening. For (i), $w^{(t)}(u, v)$ becomes $w^{(t)}(u, v) \exp(a_e)$ while $w^{(t)}(u, z)$ is unchanged. Hence $\tilde{S}_t(u)$ cancels in the ratio and

$$\frac{\tilde{P}^{(t)}(u, v)}{\tilde{P}^{(t)}(u, z)} = \frac{w^{(t)}(u, v) \exp(a_e)}{w^{(t)}(u, z)} = \exp(a_e)\, \frac{P^{(t)}(u, v)}{P^{(t)}(u, z)}.$$

For (ii), only $w^{(t)}(u, v)$ changes at u, so $\tilde{S}_t(u) = S_t(u) - w^{(t)}(u, v) + w^{(t)}(u, v) \exp(a_e)$. Dividing the numerator and denominator of $w^{(t)}(u, v) \exp(a_e)/\tilde{S}_t(u)$ by $S_t(u)$ gives the first identity. Taking the log-odds of both sides gives the second. □

Interpretation. The multiplier $\exp(a_e)$ in Lemma 2(i) depends on none of $w^{(t)}(u,v)$, $w^{(t)}(u,z)$, and $S_t(u)$. Strengthening therefore raises the ratio of a stored transition to each transition outside $\mathcal{M}_q$ from the same node by e^{a_e}. This factor is the same for relation, synonym, and document edges, whose weights lie on different scales (Appendix A). It is also independent of how many competing transitions expansion has added at u. An additive increase c would instead multiply the ratio by $1 + c/w^{(t)}(u,v)$, which depends on the original weight. By Lemma 2(ii), at a node with a single QCEdge, the strengthened probability increases monotonically with a_e and stays below one. Because the graph is undirected, both results also hold at v with u and v exchanged.

Edges that expansion adds after t_m did not exist when $\mathcal{M}_q$ was built, so none of them belongs to $\mathcal{M}_q$. Lemma 2(i) therefore applies at every node of a stored path against each newly added competing transition. Consider a path $\gamma = (v_0, \ldots, v_L)$ whose edges $\{v_i, v_{i+1}\}$ all belong to $\mathcal{M}_q$. For each i with a competing transition, choose z_i such that $w^{(t)}(v_i, z_i) > 0$ and $\{v_i, z_i\} \notin \mathcal{M}_q$. Applying Lemma 2(i) at each node and multiplying gives

$$\prod_{i=0}^{L-1} \frac{\tilde{P}^{(t)}(v_i, v_{i+1})}{\tilde{P}^{(t)}(v_i, z_i)} = \exp\Big(\theta_t \sum_{i=0}^{L-1} \hat{I}(v_i, v_{i+1} \mid q)\Big) \prod_{i=0}^{L-1} \frac{P^{(t)}(v_i, v_{i+1})}{P^{(t)}(v_i, z_i)}. \tag{12}$$

Relative to transitions outside $\mathcal{M}_q$, the transitions along γ are thus favored by the exponential factor in Eq. (12). Like the dilution in Eq. (7), this change compounds multiplicatively along the path. Since $\theta_t = g(t)^p$, the factor grows as competition around stored edges raises $g(t)$.

B.3 RMS Aggregation for Refresh

For a memory bank $\mathcal{Q}_M$ with $n = |\mathcal{Q}_M|$, we write $s_q = s_{\text{new}}(q \mid t_m)$. Each s_q lies in $[0,1]$ because it is a share of nonnegative similarity mass. Let $\bar{s} = n^{-1} \sum_{q \in \mathcal{Q}_M} s_q$ and $s_{\max} = \max_{q \in \mathcal{Q}_M} s_q$.

Lemma 3 (Properties of RMS aggregation)**.** *The staleness signal $s_{\text{mem}}(t)$ in Eq.* (4) *satisfies the following.*

(i)

$$s_{\text{mem}}(t)^2 = \bar{s}^{\,2} + n^{-1} \sum_{q \in \mathcal{Q}_M} (s_q - \bar{s})^2.$$

(ii)

$$\bar{s} \le s_{\text{mem}}(t) \le s_{\max}.$$

(iii) If m of the n queries have $s_q = a \in (0,1]$ and the others have $s_q = 0$, then

$$\bar{s} = a\,\frac{m}{n}, \qquad s_{\text{mem}}(t) = a\sqrt{\frac{m}{n}}. \tag{13}$$

Proof. By Eq. (4), $s_{\text{mem}}(t)^2 = n^{-1} \sum_q s_q^2$. Expanding $n^{-1} \sum_q (s_q - \bar{s})^2 = n^{-1} \sum_q s_q^2 - \bar{s}^{\,2}$ gives (i). The lower bound in (ii) follows from (i), and the upper bound follows from $s_q^2 \le s_{\max}^2$. Substituting the two values gives (iii). □

Interpretation. By Lemma 3(i), $s_{\text{mem}}(t)^2$ is the squared mean plus the variance across queries. When expansion changes triple retrieval for only part of the bank, s_q varies widely across queries. The mean then spreads the shift over the whole bank, whereas RMS also reflects the variance across queries. By (ii), RMS lies between the mean and the maximum. In (iii), the ratio $s_{\text{mem}}(t)/\bar{s} = \sqrt{n/m}$ grows as the shift concentrates in fewer queries. When the shift is confined to one query ($m = 1$), $s_{\text{mem}}(t) = s_{\max}/\sqrt{n}$. RMS therefore gives more weight than the mean to shifts concentrated in a subset of queries and reacts less than the maximum to an isolated query.

C Experimental Protocols and Corpus Statistics

C.1 Evaluation Details

Delayed-evidence release. For each delayed query, half of its supporting documents are assigned to the initial step, and the rest are randomly distributed across later steps. Documents shared by multiple

queries are indexed only once and enter the corpus when first released for any query. Consequently, earlier release of shared documents may increase the evidence initially available to a query and reduce the evidence arriving later. We do not adjust for these differences.

Preprocessing runs. The single-run experiments use the preprocessing run whose retrieval performance is closest to the five-run mean. All compared methods and configurations share the same run, corpus construction, and document release assignments.

C.2 Corpus Statistics

Table 3 reports the corpus and graph sizes at each expansion step on MuSiQue.

Table 3: MuSiQue corpus and graph statistics across expansion steps.

Step	Documents	Nodes	Edges
250	3,950	41,651	415,833
500	7,009	71,021	1,212,129
750	9,382	92,553	2,327,448
1000	11,656	113,297	3,784,610

D Full and Additional Experimental Results

This section reports full step-wise and block-wise results, delayed-evidence results, parameter sensitivity, and the detailed qualitative case study. The corresponding experimental protocols are defined in Appendix C.

D.1 Fixed-Query Results

Tables 4 and 5 report the complete step-wise results for the fixed-query protocol.

Table 4: **Full step-wise Recall@5 results in the fixed-query continual setting.** Best results are in bold, and second-best results are underlined.

Dataset	Step	text-embedding-large	Qwen-8B	NV-Embed-v2	KG2RAG	HippoRAG2	EviWeaver
MuSiQue	250	0.638	0.663	0.720	0.522	0.771	**0.799**
	500	0.607	0.639	0.697	0.461	0.751	**0.770**
	750	0.591	0.621	0.674	0.461	0.735	**0.772**
	1000	0.577	0.616	0.666	0.434	0.714	**0.766**
2WikiMultiHopQA	250	0.782	0.775	0.791	0.785	**0.957**	0.938
	500	0.767	0.760	0.780	0.789	0.918	**0.924**
	750	0.758	0.747	0.773	0.795	0.880	**0.912**
	1000	0.753	0.738	0.766	0.783	0.847	**0.902**
HotpotQA	250	0.896	0.894	0.946	0.824	**0.964**	0.956
	500	0.890	0.886	0.942	0.824	0.962	**0.964**
	750	0.886	0.878	0.938	0.812	0.955	**0.956**
	1000	0.884	0.878	0.936	0.806	0.955	**0.956**

Table 5: **Full step-wise answer F1 results in the fixed-query continual setting.** Best results are in bold, and second-best results are underlined.

Dataset	Step	text-embedding-large	Qwen-8B	NV-Embed-v2	KG2RAG	HippoRAG2	EraRAG	HyperGraphRAG	EviWeaver
MuSiQue	250	0.417	0.431	0.447	0.388	0.507	0.401	0.350	**0.552**
	500	0.421	0.412	0.419	0.367	0.486	0.325	0.335	**0.507**
	750	0.390	0.401	0.417	0.380	0.480	0.294	0.325	**0.497**
	1000	0.387	0.404	0.403	0.348	0.453	0.296	0.309	**0.491**
2WikiMultiHopQA	250	0.574	0.590	0.583	0.564	**0.730**	0.450	0.408	0.681
	500	0.571	0.579	0.573	0.562	0.678	0.384	0.408	**0.682**
	750	0.553	0.568	0.570	0.568	0.660	0.326	0.385	**0.674**
	1000	0.546	0.557	0.557	0.546	0.619	0.303	0.396	**0.679**
HotpotQA	250	0.688	0.693	**0.734**	0.642	0.727	0.685	0.569	0.720
	500	0.689	0.693	**0.728**	0.640	0.720	0.646	0.556	0.716
	750	0.678	0.674	**0.727**	0.625	0.708	0.612	0.556	0.715
	1000	0.682	0.675	**0.720**	0.629	**0.720**	0.551	0.551	0.710

D.2 Query-Expanding Results

The following tables report every available query-block/step pair under the query-expanding protocol. Dashes denote blocks that have not yet been introduced. Recall@5 is reported for passage-aligned retrievers, and answer F1 is reported for all methods with generation results.

Table 6: **Full block-wise results under the query-expanding continual setting on MuSiQue.** Recall@5 is reported only for methods that expose ranked passage outputs aligned with gold document annotations. Dashes indicate that the query block has not yet been introduced at the corresponding step.

(a) Passage Recall@5.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.638	0.607	0.591	0.577
	Qwen-8B	0.663	0.639	0.621	0.616
	NV-Embed-v2	0.720	0.697	0.674	0.666
	KG2RAG	0.522	0.461	0.461	0.434
	HippoRAG2	0.771 ± 0.001	0.751 ± 0.002	0.735 ± 0.001	0.714 ± 0.001
	EviWeaver	0.799 ± 0.004	0.770 ± 0.002	0.772 ± 0.003	0.766 ± 0.003
251–500	text-embedding-large	–	0.650	0.634	0.621
	Qwen-8B	–	0.659	0.643	0.642
	NV-Embed-v2	–	0.728	0.719	0.711
	KG2RAG	–	0.539	0.500	0.462
	HippoRAG2	–	0.794 ± 0.002	0.771 ± 0.001	0.758 ± 0.002
	EviWeaver	–	0.814 ± 0.004	0.783 ± 0.002	0.767 ± 0.002
501–750	text-embedding-large	–	–	0.581	0.563
	Qwen-8B	–	–	0.595	0.588
	NV-Embed-v2	–	–	0.702	0.691
	KG2RAG	–	–	0.482	0.448
	HippoRAG2	–	–	0.751 ± 0.001	0.717 ± 0.001
	EviWeaver	–	–	0.762 ± 0.002	0.726 ± 0.002
751–1000	text-embedding-large	–	–	–	0.614
	Qwen-8B	–	–	–	0.625
	NV-Embed-v2	–	–	–	0.706
	KG2RAG	–	–	–	0.453
	HippoRAG2	–	–	–	0.735 ± 0.002
	EviWeaver	–	–	–	0.786 ± 0.004

(b) Answer F1.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.417 ± 0.018	0.421 ± 0.011	0.390 ± 0.011	0.387 ± 0.008
	Qwen-8B	0.431 ± 0.004	0.412 ± 0.010	0.401 ± 0.006	0.404 ± 0.009
	NV-Embed-v2	0.447 ± 0.004	0.419 ± 0.011	0.417 ± 0.011	0.403 ± 0.009
	KG2RAG	0.388 ± 0.006	0.367 ± 0.005	0.380 ± 0.007	0.348 ± 0.010
	HippoRAG2	0.507 ± 0.004	0.486 ± 0.010	0.480 ± 0.010	0.453 ± 0.009
	HyperGraphRAG	0.350 ± 0.015	0.335 ± 0.014	0.325 ± 0.014	0.309 ± 0.015
	EraRAG	0.401 ± 0.011	0.325 ± 0.013	0.294 ± 0.008	0.296 ± 0.009
	EviWeaver	0.552 ± 0.007	0.507 ± 0.012	0.497 ± 0.010	0.491 ± 0.007
251–500	text-embedding-large	–	0.450 ± 0.009	0.447 ± 0.018	0.424 ± 0.010
	Qwen-8B	–	0.445 ± 0.012	0.435 ± 0.010	0.425 ± 0.019
	NV-Embed-v2	–	0.491 ± 0.009	0.482 ± 0.009	0.464 ± 0.010
	KG2RAG	–	0.382 ± 0.008	0.396 ± 0.010	0.359 ± 0.007
	HippoRAG2	–	0.560 ± 0.009	0.547 ± 0.007	0.520 ± 0.012
	HyperGraphRAG	–	0.357 ± 0.009	0.344 ± 0.011	0.340 ± 0.019
	EraRAG	–	0.400 ± 0.005	0.342 ± 0.011	0.341 ± 0.007
	EviWeaver	–	0.591 ± 0.010	0.551 ± 0.006	0.525 ± 0.006
501–750	text-embedding-large	–	–	0.349 ± 0.013	0.350 ± 0.012
	Qwen-8B	–	–	0.352 ± 0.014	0.346 ± 0.004
	NV-Embed-v2	–	–	0.405 ± 0.012	0.409 ± 0.006
	KG2RAG	–	–	0.351 ± 0.007	0.318 ± 0.003
	HippoRAG2	–	–	0.472 ± 0.009	0.445 ± 0.005
	HyperGraphRAG	–	–	0.282 ± 0.019	0.275 ± 0.010
	EraRAG	–	–	0.268 ± 0.010	0.276 ± 0.011
	EviWeaver	–	–	0.463 ± 0.012	0.453 ± 0.006
751–1000	text-embedding-large	–	–	–	0.435 ± 0.018
	Qwen-8B	–	–	–	0.366 ± 0.005
	NV-Embed-v2	–	–	–	0.418 ± 0.015
	KG2RAG	–	–	–	0.340 ± 0.005
	HippoRAG2	–	–	–	0.479 ± 0.009
	HyperGraphRAG	–	–	–	0.360 ± 0.019
	EraRAG	–	–	–	0.309 ± 0.014
	EviWeaver	–	–	–	0.540 ± 0.004

Table 7: **Full block-wise results under the query-expanding continual setting on 2WikiMultiHopQA.** Recall@5 is reported only for methods that expose ranked passage outputs aligned with gold document annotations. Dashes indicate that the query block has not yet been introduced at the corresponding step.

(a) Passage Recall@5.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.782	0.767	0.758	0.753
	Qwen-8B	0.775	0.760	0.747	0.738
	NV-Embed-v2	0.791	0.780	0.773	0.766
	KG2RAG	0.785	0.789	0.795	0.783
	HippoRAG2	0.957 ± 0.002	0.918 ± 0.001	0.880 ± 0.002	0.847 ± 0.001
	EviWeaver	0.938 ± 0.002	0.924 ± 0.000	0.912 ± 0.001	0.902 ± 0.001
251–500	text-embedding-large	–	0.758	0.749	0.742
	Qwen-8B	–	0.750	0.746	0.744
	NV-Embed-v2	–	0.780	0.770	0.767
	KG2RAG	–	0.803	0.792	0.791
	HippoRAG2	–	0.920 ± 0.003	0.869 ± 0.002	0.844 ± 0.001
	EviWeaver	–	0.921 ± 0.002	0.885 ± 0.004	0.879 ± 0.003
501–750	text-embedding-large	–	–	0.737	0.733
	Qwen-8B	–	–	0.735	0.731
	NV-Embed-v2	–	–	0.757	0.748
	KG2RAG	–	–	0.761	0.774
	HippoRAG2	–	–	0.907 ± 0.002	0.864 ± 0.003
	EviWeaver	–	–	0.895 ± 0.001	0.870 ± 0.001
751–1000	text-embedding-large	–	–	–	0.763
	Qwen-8B	–	–	–	0.759
	NV-Embed-v2	–	–	–	0.782
	KG2RAG	–	–	–	0.804
	HippoRAG2	–	–	–	0.890 ± 0.002
	EviWeaver	–	–	–	0.902 ± 0.002

(b) Answer F1.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.574 ± 0.013	0.571 ± 0.010	0.553 ± 0.011	0.546 ± 0.014
	Qwen-8B	0.590 ± 0.016	0.579 ± 0.012	0.568 ± 0.007	0.557 ± 0.011
	NV-Embed-v2	0.583 ± 0.009	0.573 ± 0.003	0.570 ± 0.013	0.557 ± 0.007
	KG2RAG	0.564 ± 0.004	0.562 ± 0.007	0.568 ± 0.007	0.546 ± 0.010
	HippoRAG2	0.730 ± 0.004	0.678 ± 0.010	0.660 ± 0.007	0.619 ± 0.009
	HyperGraphRAG	0.408 ± 0.005	0.408 ± 0.014	0.385 ± 0.006	0.396 ± 0.010
	EraRAG	0.450 ± 0.006	0.384 ± 0.006	0.326 ± 0.007	0.303 ± 0.006
	EviWeaver	0.681 ± 0.010	0.682 ± 0.007	0.674 ± 0.011	0.679 ± 0.005
251–500	text-embedding-large	–	0.575 ± 0.011	0.561 ± 0.008	0.569 ± 0.021
	Qwen-8B	–	0.590 ± 0.010	0.591 ± 0.016	0.582 ± 0.007
	NV-Embed-v2	–	0.606 ± 0.009	0.592 ± 0.009	0.601 ± 0.014
	KG2RAG	–	0.610 ± 0.005	0.590 ± 0.001	0.593 ± 0.009
	HippoRAG2	–	0.710 ± 0.012	0.667 ± 0.015	0.648 ± 0.009
	HyperGraphRAG	–	0.439 ± 0.016	0.420 ± 0.016	0.427 ± 0.008
	EraRAG	–	0.442 ± 0.012	0.371 ± 0.007	0.325 ± 0.011
	EviWeaver	–	0.728 ± 0.005	0.696 ± 0.007	0.685 ± 0.004
501–750	text-embedding-large	–	–	0.562 ± 0.007	0.555 ± 0.012
	Qwen-8B	–	–	0.559 ± 0.007	0.568 ± 0.011
	NV-Embed-v2	–	–	0.580 ± 0.012	0.575 ± 0.022
	KG2RAG	–	–	0.591 ± 0.003	0.602 ± 0.006
	HippoRAG2	–	–	0.703 ± 0.004	0.667 ± 0.006
	HyperGraphRAG	–	–	0.416 ± 0.010	0.405 ± 0.017
	EraRAG	–	–	0.381 ± 0.007	0.357 ± 0.009
	EviWeaver	–	–	0.709 ± 0.009	0.668 ± 0.004
751–1000	text-embedding-large	–	–	–	0.592 ± 0.010
	Qwen-8B	–	–	–	0.619 ± 0.010
	NV-Embed-v2	–	–	–	0.616 ± 0.009
	KG2RAG	–	–	–	0.623 ± 0.005
	HippoRAG2	–	–	–	0.676 ± 0.010
	HyperGraphRAG	–	–	–	0.412 ± 0.010
	EraRAG	–	–	–	0.355 ± 0.009
	EviWeaver	–	–	–	0.736 ± 0.004

Table 8: **Full block-wise results under the query-expanding continual setting on HotpotQA.** Recall@5 is reported only for methods that expose ranked passage outputs aligned with gold document annotations. Dashes indicate that the query block has not yet been introduced at the corresponding step.

(a) Passage Recall@5.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.896	0.890	0.886	0.884
	Qwen-8B	0.894	0.886	0.878	0.878
	NV-Embed-v2	0.946	0.942	0.938	0.936
	KG2RAG	0.824	0.824	0.812	0.806
	HippoRAG2	0.964 ± 0.001	0.962 ± 0.002	0.955 ± 0.001	0.955 ± 0.001
	EviWeaver	0.956 ± 0.002	0.964 ± 0.000	0.956 ± 0.001	0.956 ± 0.002
251–500	text-embedding-large	–	0.918	0.912	0.908
	Qwen-8B	–	0.912	0.908	0.908
	NV-Embed-v2	–	0.958	0.956	0.954
	KG2RAG	–	0.874	0.856	0.838
	HippoRAG2	–	0.982 ± 0.001	0.970 ± 0.000	0.962 ± 0.001
	EviWeaver	–	0.982 ± 0.003	0.974 ± 0.002	0.976 ± 0.002
501–750	text-embedding-large	–	–	0.876	0.878
	Qwen-8B	–	–	0.890	0.890
	NV-Embed-v2	–	–	0.938	0.936
	KG2RAG	–	–	0.848	0.844
	HippoRAG2	–	–	0.958 ± 0.002	0.956 ± 0.000
	EviWeaver	–	–	0.956 ± 0.001	0.958 ± 0.001
751–1000	text-embedding-large	–	–	–	0.876
	Qwen-8B	–	–	–	0.884
	NV-Embed-v2	–	–	–	0.946
	KG2RAG	–	–	–	0.838
	HippoRAG2	–	–	–	0.953 ± 0.001
	EviWeaver	–	–	–	0.956 ± 0.001

(b) Answer F1.

Query Block	Method	Step 250	Step 500	Step 750	Step 1000
1–250	text-embedding-large	0.688 ± 0.008	0.689 ± 0.007	0.678 ± 0.007	0.682 ± 0.007
	Qwen-8B	0.693 ± 0.009	0.693 ± 0.011	0.674 ± 0.007	0.675 ± 0.012
	NV-Embed-v2	0.734 ± 0.006	0.728 ± 0.006	0.727 ± 0.016	0.720 ± 0.008
	KG2RAG	0.642 ± 0.001	0.640 ± 0.006	0.625 ± 0.002	0.629 ± 0.003
	HippoRAG2	0.727 ± 0.003	0.720 ± 0.004	0.708 ± 0.005	0.720 ± 0.003
	HyperGraphRAG	0.569 ± 0.009	0.556 ± 0.006	0.556 ± 0.019	0.551 ± 0.012
	EraRAG	0.685 ± 0.007	0.646 ± 0.004	0.612 ± 0.008	0.551 ± 0.009
	EviWeaver	0.720 ± 0.002	0.716 ± 0.000	0.715 ± 0.001	0.710 ± 0.001
251–500	text-embedding-large	–	0.714 ± 0.004	0.717 ± 0.009	0.714 ± 0.002
	Qwen-8B	–	0.726 ± 0.003	0.713 ± 0.006	0.716 ± 0.004
	NV-Embed-v2	–	0.743 ± 0.007	0.749 ± 0.006	0.751 ± 0.003
	KG2RAG	–	0.701 ± 0.004	0.687 ± 0.005	0.649 ± 0.005
	HippoRAG2	–	0.775 ± 0.004	0.763 ± 0.004	0.747 ± 0.008
	HyperGraphRAG	–	0.570 ± 0.013	0.569 ± 0.011	0.564 ± 0.008
	EraRAG	–	0.681 ± 0.005	0.623 ± 0.012	0.591 ± 0.008
	EviWeaver	–	0.772 ± 0.003	0.765 ± 0.003	0.770 ± 0.003
501–750	text-embedding-large	–	–	0.688 ± 0.009	0.688 ± 0.009
	Qwen-8B	–	–	0.695 ± 0.011	0.718 ± 0.010
	NV-Embed-v2	–	–	0.744 ± 0.008	0.746 ± 0.017
	KG2RAG	–	–	0.680 ± 0.004	0.666 ± 0.003
	HippoRAG2	–	–	0.757 ± 0.004	0.762 ± 0.004
	HyperGraphRAG	–	–	0.519 ± 0.009	0.515 ± 0.005
	EraRAG	–	–	0.653 ± 0.002	0.627 ± 0.010
	EviWeaver	–	–	0.750 ± 0.002	0.764 ± 0.001
751–1000	text-embedding-large	–	–	–	0.645 ± 0.005
	Qwen-8B	–	–	–	0.692 ± 0.016
	NV-Embed-v2	–	–	–	0.680 ± 0.009
	KG2RAG	–	–	–	0.629 ± 0.007
	HippoRAG2	–	–	–	0.687 ± 0.004
	HyperGraphRAG	–	–	–	0.516 ± 0.011
	EraRAG	–	–	–	0.568 ± 0.007
	EviWeaver	–	–	–	0.703 ± 0.002

Table 9: **Full delayed-evidence results under permuted release order.**

(a) MuSiQue, Recall@5.

Ratio	Method	500	750	1000	Avg.
0%	HippoRAG2	0.7343	0.7327	0.7233	0.7301
	EviWeaver	0.7490	0.7450	0.7510	**0.7483**
25%	HippoRAG2	0.7167	0.7367	0.7227	0.7254
	EviWeaver	0.7290	0.7430	0.7460	**0.7393**
50%	HippoRAG2	0.7073	0.7343	0.7213	0.7210
	EviWeaver	0.7230	0.7453	0.7410	**0.7364**
75%	HippoRAG2	0.7050	0.7323	0.7227	0.7200
	EviWeaver	0.7127	0.7547	0.7387	**0.7354**
100%	HippoRAG2	0.7033	0.7323	0.7253	0.7203
	EviWeaver	0.7230	0.7497	0.7463	**0.7397**

(b) 2WikiMultiHopQA, Recall@5.

Ratio	Method	500	750	1000	Avg.
0%	HippoRAG2	0.9220	0.8780	0.8490	0.8830
	EviWeaver	0.9250	0.9130	0.9010	**0.9130**
25%	HippoRAG2	0.8860	0.8800	0.8490	0.8717
	EviWeaver	0.8940	0.8940	0.8840	**0.8907**
50%	HippoRAG2	0.8680	0.8830	0.8530	0.8680
	EviWeaver	0.8720	0.9040	0.8830	**0.8863**
75%	HippoRAG2	0.8390	0.8870	0.8510	0.8590
	EviWeaver	0.8330	0.8970	0.8780	**0.8693**
100%	HippoRAG2	0.8260	0.8900	0.8530	0.8563
	EviWeaver	0.8160	0.9030	0.8780	**0.8657**

(c) MuSiQue, answer F1.

Ratio	Method	500	750	1000	Avg.
0%	HippoRAG2	0.5452	0.5429	0.5361	0.5414
	EviWeaver	0.5518	0.5458	0.5520	**0.5499**
25%	HippoRAG2	0.5431	0.5482	0.5353	0.5422
	EviWeaver	0.5532	0.5564	0.5570	**0.5555**
50%	HippoRAG2	0.5452	0.5494	0.5473	0.5473
	EviWeaver	0.5522	0.5510	0.5528	**0.5520**
75%	HippoRAG2	0.5588	0.5677	0.5465	0.5577
	EviWeaver	0.5563	0.5725	0.5523	**0.5604**
100%	HippoRAG2	0.5431	0.5526	0.5513	0.5490
	EviWeaver	0.5662	0.5618	0.5544	**0.5608**

(d) 2WikiMultiHopQA, answer F1.

Ratio	Method	500	750	1000	Avg.
0%	HippoRAG2	0.6703	0.6689	0.6189	0.6527
	EviWeaver	0.6792	0.6873	0.6698	**0.6788**
25%	HippoRAG2	0.6523	0.6606	0.6238	0.6456
	EviWeaver	0.6569	0.6811	0.6508	**0.6629**
50%	HippoRAG2	0.6469	0.6588	0.6345	0.6467
	EviWeaver	0.6530	0.6668	0.6459	**0.6552**
75%	HippoRAG2	0.6297	0.6751	0.6245	0.6431
	EviWeaver	0.6316	0.6642	0.6518	**0.6492**
100%	HippoRAG2	0.6176	0.6804	0.6285	0.6422
	EviWeaver	0.6205	0.6717	0.6390	**0.6437**

D.3 Delayed-Evidence Results

Table 9 reports full-chain Recall@5 and answer F1 at every expansion step across delayed-query ratios under the permuted release order. The 0% rows provide the corresponding no-delay controls. Step 250 is the construction step and is excluded. Avg. is computed over Steps 500–1000.

D.4 Parameter Sensitivity

Adaptive controller. Both parameters are robust. Retrieval quality varies by 0.3 points over $\tau_c \in [0.20, 0.40]$ and 0.3 points over $p \in [0.5, 3.0]$, and every setting exceeds HippoRAG2. Setting $p = 0$ fixes the strength at 1 regardless of how much dilution has occurred and costs 0.7 points, which places the gain in scaling the strengthening to the measured dilution rather than in strengthening at all. Lowering τ_c refreshes more often and scores marginally higher, but each refresh runs the decomposition again.

Memory construction. The QCEdge memory size B and the restart probability α_{bw} govern path-memory capacity and backward contribution tracing rather than the controller, so we vary them on MuSiQue under the fixed-query protocol. Retrieval quality changes by at most 1.1 points over $B \in [10, 60]$ and 0.2 points over $\alpha_{\mathrm{bw}} \in [0.2, 0.6]$. $B = 30$ is the smallest memory that reaches the plateau, and larger memories bring no further gain.

Table 10: **Sensitivity of the adaptive controller.** Selected configurations in bold.

(a) Refresh threshold τ_c with $p = 2$.

τ_c	500	750	1000	Avg. ↑	Stability ↑
0.20	0.7996	0.8259	0.8133	0.8129	+0.0137
0.25	0.7996	0.8259	0.8133	0.8129	+0.0137
0.30	0.7992	0.8243	0.8117	0.8117	+0.0125
0.35	**0.7980**	**0.8249**	**0.8151**	**0.8127**	**+0.0172**
0.40	0.7975	0.8212	0.8111	0.8100	+0.0136
HippoRAG2	0.7908	0.8086	0.7870	0.7955	−0.0037

(b) strengthening exponent p with $\tau_c = 0.35$.

p	500	750	1000	Avg. ↑	Stability ↑
0.0	0.7971	0.8213	0.7993	0.8059	+0.0022
0.5	0.7971	0.8236	0.8071	0.8093	+0.0100
1.0	0.7980	0.8253	0.8098	0.8111	+0.0118
1.5	0.7977	0.8248	0.8138	0.8121	+0.0162
2.0	**0.7980**	**0.8249**	**0.8151**	**0.8127**	**+0.0172**
2.5	0.7969	0.8238	0.8164	0.8124	+0.0195
3.0	0.7962	0.8235	0.8168	0.8122	+0.0206
HippoRAG2	0.7908	0.8086	0.7870	0.7955	−0.0037

Table 11: **Sensitivity of path-memory construction.** Selected configurations in bold.

(a) QCEdge memory size B.

B	500	750	1000	Avg. ↑	Stability ↑
10	0.7660	0.7560	0.7470	0.7563	−0.0527
20	0.7660	0.7590	0.7530	0.7593	−0.0467
30	**0.7690**	**0.7670**	**0.7640**	**0.7667**	**−0.0357**
40	0.7700	0.7680	0.7640	0.7673	−0.0357
50	0.7680	0.7680	0.7570	0.7643	−0.0427
60	0.7660	0.7680	0.7590	0.7643	−0.0407

(b) restart probability α_{bw} with $B = 30$.

α_{bw}	500	750	1000	Avg. ↑	Stability ↑
0.2	0.7710	0.7710	0.7630	0.7683	−0.0367
0.3	0.7680	0.7670	0.7660	0.7670	−0.0337
0.4	0.7680	0.7690	0.7650	0.7673	−0.0347
0.5	**0.7690**	**0.7670**	**0.7640**	**0.7667**	**−0.0357**
0.6	0.7700	0.7670	0.7620	0.7663	−0.0377

D.5 HotpotQA Results

HotpotQA exhibits little retrieval degradation under corpus expansion. HippoRAG2 and EviWeaver both maintain near-saturated Recall@5. Tables 4, 5 and 8 report the full continual results. Table 12 additionally reports the ablation results.

D.6 Case Study Details

Population. We analyze 1,228 query–gold passage pairs on MuSiQue with a delayed-query ratio of 0% and 2WikiMultiHopQA with a ratio of 25%.

Criteria. A gold passage is *lost* if HippoRAG2 ranks it outside the top-5 at any of steps 500, 750, and 1000, and *kept* otherwise. A lost passage is *recovered* if EviWeaver ranks it in the top-5 at any step where HippoRAG2 misses it. Table 13 lists the number of pairs in each group.

Structural measures. Entry entities are defined from the seeds of the original query on the step-250 graph. When several shortest paths exist, we count the distinct entities one hop before the passage across these paths. Entry degree is the degree of these entry entities on the step-250 and step-1000 graphs. Entity degree is measured on the step-250 graph. The QCEdge column reports the degree of entities that QCEdges touch, and the Other column reports the degree of the remaining neighbors of the gold passage.

Step-wise cases. Figure 7 shows the three cases of Figure 6 side by side for HippoRAG2 and EviWeaver at steps 250 and 1000. Within a row, comparing the first and third columns shows how expansion changes HippoRAG2, and comparing the third and fourth columns shows how the two methods differ on the same graph. In all three cases, HippoRAG2 misses the gold passage at step 1000, while EviWeaver recovers it on the same graph through path memory.

The Entrance case shows the role of paths leading into an entry entity. At step 250, EviWeaver stores QCEdges from the seeds to an entry entity. At step 1000, even as the area around the entry entities grows crowded, a QCEdge path through a less crowded entry entity reaches the passage.

The Bridge case shows the role of a bridge entity shared by gold passages. At step 250, EviWeaver reaches this entity through decomposition and stores the paths around it as QCEdges. At step 1000, these QCEdges retrieve the passage that HippoRAG2 misses.

Table 12: **Ablation results on HotpotQA.** Recall@5 is averaged over the fixed-query and delayed-evidence settings.

Step	EviWeaver	w/o QCEdge	w/o QD	w/o Refresh	w/o Adaptive Strength
250	0.723	0.723	0.729	0.723	0.723
500	0.966	0.966	0.965	0.966	0.966
750	0.957	0.957	0.959	0.957	0.957
1000	0.955	0.956	0.953	0.956	0.956

Table 13: Number of query–gold passage pairs in each group of Figure 6d.

By HippoRAG2		Lost, by EviWeaver		Recovered, by situation		
Kept	Lost	Recovered	Unrecovered	Entrance	Bridge	Pool
896	332	64	268	19	7	38

The Pool case shows the role of low-degree entities around the passage. At step 250, EviWeaver finds seeds close to the passage through decomposition and, from them, stores QCEdges among low-degree entities around the passage. At step 1000, these QCEdges recover the passage even though its entry entities grow crowded.

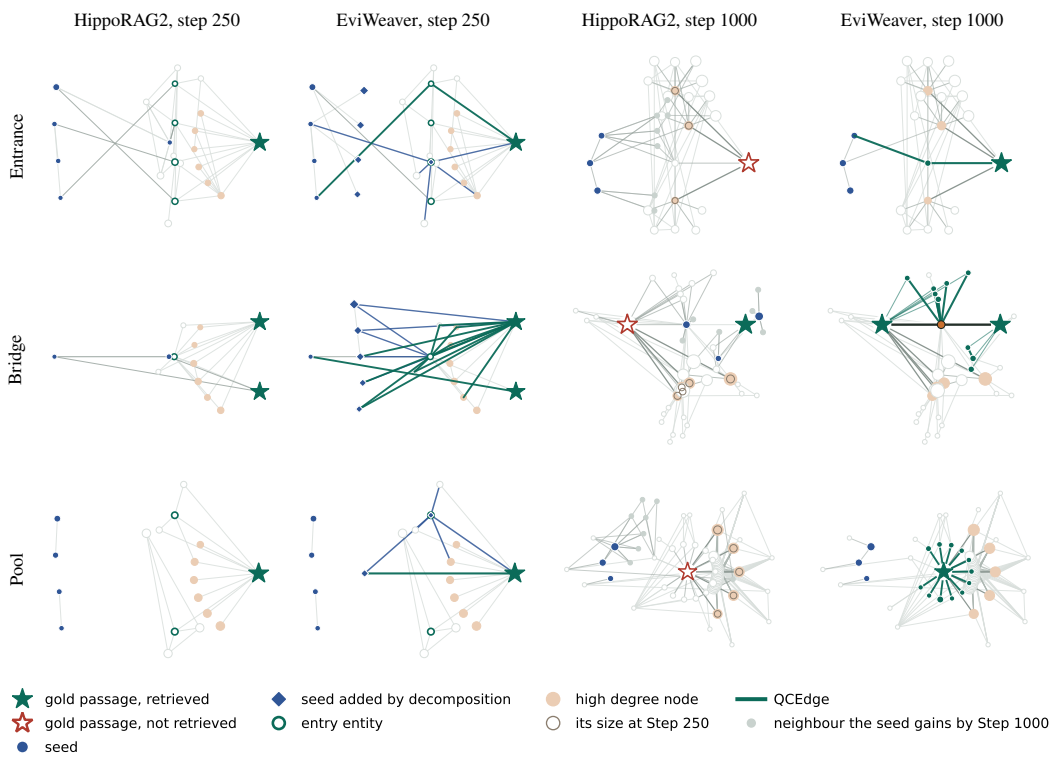


Figure 7: **Step-wise view of the cases in Figure 6.** Rows are situations and columns are methods and steps. Node size is proportional to degree on a common scale within each row.

E Computational Efficiency

Computational resources. Local experiments run on a server with two Intel Xeon Gold 6526Y CPUs, 188 GB RAM, and four NVIDIA RTX A6000 GPUs with 48 GB VRAM each. GPT-4o-mini is accessed through an external API.

E.1 Runtime and LLM Usage

We compare the retrieval costs of EviWeaver and HippoRAG2 in the delayed-evidence setting. Table 14 reports average costs per query across the four evaluation steps, including initial memory construction and subsequent strengthening or refresh. Local computation time excludes LLM API calls and is measured on the largest graph (step 1000).

Table 14: Average retrieval costs per query in the delayed-evidence setting.

Method	LLM calls	Input tokens	Output tokens	Local compute (s)
HippoRAG2	1.04	2,939	61	2.27
EviWeaver	2.55	6,140	167	2.40

EviWeaver uses the same index as HippoRAG2 and therefore incurs no additional indexing cost. Memory construction reuses the forward PPR of retrieval and adds only 0.07 s of backward tracing. Additional LLM usage comes from bridge-aware seed construction when a query first arrives or its memory is refreshed. Strengthening uses the same LLM filtering as HippoRAG2 without repeating query decomposition.

E.2 Memory Footprint

Each query stores at most $B = 30$ QCEdges, represented by their endpoint identifiers and normalized importance scores. QCEdge storage scales as $O(|\mathcal{Q}|B)$ and does not grow with graph size for a fixed query bank. Table 15 reports the serialized QCEdge storage under query expansion. It remains below 2 MB for 1,000 queries. These values exclude the shared graph, embedding indices, and other caches.

Table 15: Serialized QCEdge-memory footprint under query expansion.

Step	Queries	Stored QCEdges	Size (KB)
250	250	7,500	459.7
500	500	15,000	917.2
750	750	22,500	1,375.1
1000	1000	30,000	1,841.7

F Prompt Templates

We show condensed versions of the two prompts used for bridge-aware seed construction. Few-shot examples and output schemas are omitted here and provided in our repository.[1]

F.1 Atomic Question Extraction

The first prompt extracts atomic questions and describes their role in the reasoning chain.

```
You are an expert in multi-hop question understanding. Extract 1-2 atomic factual bridge questions from a complex multi-hop
question and produce structured descriptions of their role in the reasoning chain. Each atomic question must be answerable by a
 single KG triple.

# Task Instructions

1. Identify candidate noun phrases verbatim, including full relative clauses.
2. Select early-hop anchors that the rest of the question logically depends on.
3. Convert selected anchors into minimal atomic bridge questions answerable by one KG triple, beginning with "Who", "What", or
"Which".
4. For each bridge question, generate Info_obtained, Role_in_chain, and Remaining_subproblems.
5. Verify that there is no noun-phrase paraphrasing, no invented entities or inferred facts, no chain-of-thought, and that the
JSON structure matches exactly.

# Output Format
[Omitted; see repository]

# Examples
[Omitted; see repository]

# Input
Question: <<ORIGINAL_QUERY>>

Return JSON only.
```

F.2 Query Decomposition

The second prompt rewrites the query with entities from the triples retrieved for the atomic questions and decomposes it into sub-queries.

```
You are an expert decomposition agent for multi-hop factual reasoning. Given the atomic bundle, descriptions, bridge triples
retrieved by the atomic questions, and original query, produce the simplest ordered list of factual sub-questions that
reconstructs the reasoning path.

# Task Instructions

1. Resolve ambiguous entities using atomic questions and retrieved triples.
2. Replace complex noun phrases with resolved concrete entities to form a reduced query.
3. Decompose the reduced query into a short ordered chain of factual sub-questions.
4. Ensure that the final sub-question matches the meaning of the reduced query.
5. Use only verbatim noun phrases from the original query or grounded entities from retrieved triples.
```

[1] https://anonymous.4open.science/r/EVIWEAVER-A367

```
6. Do not introduce new entities, reuse atomic bridge questions, or output chain-of-thought.

# Output Format
[Omitted; see repository]

# Examples
[Omitted; see repository]

# Input
Context bundle: <<ATOMIC_AND_TRIPLES>>
Original query: <<ORIGINAL_QUERY>>

Return JSON only.
```