Graph Tokens before Graph Propagation: Atom-Keyed Retrieval Interfaces for Graph Foundation Models

Jiayi Qiu [1]

Abstract

We argue, in the context of single-document long-context QA, that the unit on which a graph-to-LLM retriever *ranks* and the unit it *reads* from benefit from being different objects, joined by a deterministic source-of map. Concretely, we treat each chunk's atomic Davidsonian proposition as a graph node; the source-of relation (atom $\rightarrow$ source chunk) as a deterministic bipartite map to the reader payload; and entity-mentions, scene index, time anchor, and edge-hints as edge-schema primitives. Our pipeline, Atom-Keyed Chunk Retrieval (AKCR), ranks atoms by query cosine, maps top atoms back to their source chunks, and feeds those chunks (not the atoms) to the reader.

We evaluate on full-book NarrativeQA (40 books, 1,169 questions). Our main empirical claim is same-reader: at fixed retrieval budget $b = 24$K chars, AKCR closes a substantial fraction of the gap to the same-reader long-context oracle, and outperforms a chunk-cosine retrieval baseline at the same budget, though it does not exceed the same-reader oracle when the oracle is reliable. With GLM-4.7 as reader, AKCR scores 0.5543 vs. a 0.6450 full-book oracle (gap -9.07pp). On the GLM-5.1 reader, the same-reader comparison goes the other way ($+13.6$pp over the full-book oracle), but we treat this as a diagnostic of long-context degradation in GLM-5.1 rather than a retrieval-vs-context win. A scoring-rule ablation finds atom-best max-pooling the strongest of the rules we tested; alternatives degrade or tie within run-to-run noise. Two negative results constrain the position: in our single-book candidate pool, a learned typed-edge propagator (LearnRet) drives $\mathrm{ppr}_\alpha \rightarrow 0.95$, effectively reducing to the dense seed; and feeding atoms directly to the reader plateaus near 0.46 across four formats and two extractors.

As a deployment-style observation we report that AKCR with Claude Opus 4.7 at $b{=}24$K reaches 0.6721 ± 0.007 over three reruns, comparable to the GLM-4.7 full-book oracle (0.6450); we are explicit that this is a cross-reader comparison and not evidence that retrieval beats full context under a matched reader. Run-to-run variance on LLM-judged accuracy is non-trivial in this regime (an AKCR v1 GLM-5.1 four-run sample, three full runs and one partial run, gave $\mathrm{SD} \approx 5$pp); we report multi-run means where available and treat sub-3pp single-run gaps as inconclusive. We do not claim graph propagation is universally unhelpful; we suggest that, on long single-document narrative QA, the choice of graph token is a useful design lever to study before adding propagation.

1. Introduction

The GFM call-for-papers calls out two themes that we address simultaneously: **graph learning with LLMs** ("GFMs that interface with language models and retrieval") and **new GFMs** ("graph tokenization, structural and positional encodings"). We take a narrow but concrete position at their intersection: in single-document long-context QA, the unit a graph-to-LLM retriever *ranks on* and the unit the reader *reads from* should be different objects, joined by a deterministic source-of map.

Standard RAG (Karpukhin et al., 2020) treats a chunk – a contiguous span of text – as both retrieval unit and reader unit. Graph-augmented variants such as GraphRAG, HippoRAG, HippoRAG 2, and RAPTOR (Edge et al., 2024; Gutiérrez et al., 2024; Jiménez Gutiérrez et al., 2025; Sarthi et al., 2024) layer entity/relation graphs, passage graphs, or hierarchical summaries on top of chunks and then retrieve or propagate over the structure. The closest prior work in spirit is GraphToken (Perozzi et al., 2024), which learns an encoding function that inserts structured graph information into LLM prompts for graph-reasoning tasks. We share the

[1]The University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, United States. Correspondence to: Jiayi Qiu <arthurq@ad.unc.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

premise that graph structure should be exposed to the language model interface, but use graph tokens differently: our tokens are not learned prompt prefixes and are not the reader payload; they are atom-level retrieval keys that deterministically map back to unmodified text chunks. Our position is that, at least for long single-document narrative QA, the structured ranking key and the natural-language content the LLM composes from benefit from being represented as different units.

Why this is a Graph Foundation Models problem. We treat each chunk's set of atomic Davidsonian propositions (Davidson, 1967) as the local nodes of a per-document graph. Each atom carries an entity-mention list, a scene index, a time anchor, and a few edge-hint relations (entity overlap, shared subject, scene co-occurrence, inferred causal/temporal links). The source-of relation $\sigma : t \to c$ that maps an atom t back to its source chunk c is a deterministic bipartite map between the graph layer and a language-payload layer. Most prior graph-to-LLM systems decide on a graph topology and a propagation operator first, and treat the choice of *node* (chunk, entity, or proposition) as a downstream tokenizer detail. We invert that ordering: we fix a particular node choice (atomic propositions), make the bipartite σ explicit, and ask whether a learned propagator over the typed atom graph adds anything once the dense atom seed already concentrates probability mass on the answer region. This recasts the design question for graph-to-LLM interfaces as "what should the graph token be?" before "which propagator do we use?"

We support this position with full-book NarrativeQA (Kočiský et al., 2018) evidence (40 books, 1,169 questions). Our pipeline, AKCR (Atom-Keyed Chunk Retrieval), extracts $89{,}575$ atomic propositions over $4{,}929$ chunks via a 3-pass GLM-4.7 recipe and ranks atoms against the query in the BGE-M3 embedding space (Chen et al., 2024). Top atoms are mapped back to source chunks via σ and the chunks are fed to the reader. Our main empirical claim is same-reader (§3.2). With GLM-4.7 as both retrieval reader and oracle reader, HACS-AKCR at $b = 24$K chars scores 0.5543 versus a 0.6450 full-book oracle (gap of -9.07pp). At the same budget, AKCR is above a DOS+chunks chunk-cosine baseline (0.5227) under the same reader, suggesting that the gap closed against the full-context oracle is not explained by the chunk-cosine baseline alone. With GLM-5.1 the same-reader comparison flips (0.5672 AKCR vs 0.4311 oracle on the full $\sim$78K book); we treat that flip as a diagnostic of long-context degradation in GLM-5.1, not as evidence that retrieval beats full context.

A scoring-rule ablation (HACS, MAR-sum, MAR-lse, ACRP) suggests atom-best max-pooling is the strongest reduction among those we tried; alternatives degrade or tie within run-to-run noise. Two negative results constrain the position: a fully learned typed-edge propagator (LearnRet, APPNP closed-form (Klicpera et al., 2019), 16 trainable parameters) drives $\mathrm{ppr}_\alpha \to 0.95$ in our single-book candidate pool, reducing the propagation update to the dense seed; and feeding atoms themselves to the reader plateaus near 0.46 across four formats and two extractors.

As a separate, deployment-style observation we note that with Claude Opus 4.7 as the retrieval reader, HACS-AKCR at $b = 24$K reaches 0.6721 ± 0.007 over three reruns, comparable to the GLM-4.7 full-book oracle (0.6450). We treat this only as a cross-reader cost-quality data point: a stronger reader on $\sim$30% of the book reaching the same judge-acc as a weaker reader on the full book. It is not evidence about retrieval vs. full context under a matched reader.

Contributions. We (i) recast AKCR as a graph-foundation tokenization choice rather than a RAG pipeline, with explicit graph-node, edge-schema, and bipartite-payload descriptions; (ii) report same-reader retrieval quality and oracle gaps on NarrativeQA, where AKCR closes a substantial fraction of the gap to the same-reader long-context oracle while not exceeding it when the oracle is reliable; (iii) give a scoring-rule ablation suggesting atom-best max-pooling is the strongest reduction in this setting; (iv) report two negative results (learned propagation self-disable, atoms-as-content plateau) that constrain when atomic tokenization helps and when graph propagation adds value; and (v) document non-trivial API run-to-run variance for LLM-judged QA accuracy (SD $\approx$ 5pp in a GLM-5.1 reader sample), with a recommendation to default to multi-run reporting and treat sub-3pp single-run gaps as inconclusive.

2. The Position

Thesis (bounded). *For long single-document narrative QA, our evidence suggests that the choice of graph token is a useful design choice to study before adding propagation. In our setting, using atomic Davidsonian propositions as ranking keys and source chunks as reader payloads, joined by a deterministic source-of map, gives a better budget–accuracy trade than chunk-cosine retrieval at the same budget, and recovers a substantial fraction of full-context oracle accuracy without exceeding it under a matched, well-behaved reader. We do not generalise to multi-hop encyclopedic QA, schema-grounded KGQA, or non-narrative long documents.*

Why role-decoupling can help here. A long document of C chunks admits two retrieval geometries: one d-dimensional vector per chunk (chunk-RAG), or roughly 17 vectors per chunk, one per atomic proposition (atom-RAG). Under the former, the chunk's score is the *average* of its propositions: one relevant fact buried in twenty unrelated ones is ranked at the average. Under the latter, the chunk's

score is the *best* of its propositions, because we rank atoms and only afterwards collapse to source chunks via σ. The geometry is reminiscent of ColBERT-v2 multi-vector retrieval (Santhanam et al., 2022), with the difference that our multi-vectors are LLM-emitted Davidsonian propositions (typed by predicate, anchored by time and scene), not transformer token vectors.

How AKCR differs from graph-RAG baselines. AKCR is not intended as a replacement for HippoRAG-style memory graphs or RAPTOR-style hierarchical summaries. Those systems are designed to construct and traverse a broader knowledge structure. AKCR tests a smaller interface question: before adding a propagator, does a finer graph token give a better ranking key for the same reader payload? This is why our primary baseline is matched-budget chunk-cosine retrieval, and why the negative propagation result below should be read as conditional on our single-book candidate pool and edge schema, not as a claim that established graph-RAG systems are unnecessary.

Why both endpoints can fail. A chunk is a coarse graph token: it averages multiple propositions and discourse roles. A flat atom list is a coarse reader payload: decontextualization (Choi et al., 2021) strips the narrative voice the reader composes from. We treat the atom as the graph token, the source chunk as the language payload, and σ as the deterministic bipartite map between them. We do not argue this is the right decomposition for graph-to-LLM interfaces in general; we argue it is a practical interface choice in the single-document long-context regime, and one whose failure cases (atoms-as-content; learned propagation on a saturated dense seed) are themselves informative about where graph-side operations might add value.

3. Empirical Evidence

3.1. Setup

We evaluate on full-book NarrativeQA (40 books, 4,929 GLM-4.7-segmented chunks of $\sim$800 chars, 1,169 questions). Let the document be a sequence of chunks $C = \{c_i\}_{i=1}^{m}$. Atomic memory is extracted once, query- independently, by a 3-pass GLM-4.7 pipeline (DEC-PROP): (1) **Decontextualize**, replacing pronouns with antecedent surface forms; (2) **Davidsonian extract**, emitting atomic propositions with explicit `subject`, `predicate`, `object`, `time_anchor`, `scene_index`, `salience`, and `edge_hints`; (3) **Faithfulness + coverage critic**, dropping hallucinations and adding missing atoms. Output: 89,575 atoms (mean $\sim$17/chunk), $\sim 1.5\times$ the original chunk char-count, $\sim$24 minutes at 200-way concurrency, one-time cost $\sim$¥150.

The graph interface is explicit. For each atom instance t_{ij} emitted from chunk c_i, the source map is $\sigma(t_{ij}) = c_i$; atoms from different chunks remain distinct even when they express similar events, so repeated concepts can return multiple source chunks rather than being merged. We add typed atom-atom edges for entity overlap, shared subject, scene co-occurrence, and LLM-emitted causal/temporal hints. The reader never sees this graph as a payload in AKCR; it sees only source chunks selected through σ.

For each query, AKCR embeds the query and atoms with frozen BGE-M3, restricted to the source story; takes top-K atoms ($K = 200$); scores a chunk by its best selected atom for v1, $S(c) = \max_{t:\sigma(t)=c} \cos(q, t)$, or by the HACS interpolation in §3.3; deduplicates chunks while preserving the best atom rank; and packs source chunks sorted by document order into the reader's character budget. Three readers are evaluated: Claude Opus 4.7, GLM-5.1, and GLM-4.7. The judge is GLM-4.7 in pairwise reference-grounded mode.

3.2. Same-reader retrieval quality and oracle gap

We organise the empirical evidence into three blocks: (1) same-reader comparisons against the full-context oracle and against a chunk-cosine retrieval baseline at matched budget; (2) a budget sweep against the chunk baseline under the GLM-5.1 reader; and (3) a cross-reader deployment-style note. Table 1 groups the same-reader rows (BLOCK A) and a chunk-cosine baseline (BLOCK B); the cross-reader observation is summarised in §3.2 and shown as a separate block (BLOCK C).

Block A: same-reader retrieval vs. full-context oracle. At $b = 24{,}000$ chars, $n = 1{,}169$, with GLM-4.7 as both retrieval reader and oracle reader, HACS-AKCR scores 0.5543 versus a 0.6450 full-book oracle: a gap of -9.07pp. On the $n = 149$ subset where we also have an Opus 4.7 oracle, AKCR scores 0.7338 versus 0.8322, a gap of -9.84pp. We do not claim AKCR closes this gap; it does not. AKCR recovers a fraction of full-context performance (about 86% of GLM-4.7 oracle, about 88% of Opus oracle on that subset) at $\sim$30% of the input cost. With the GLM-5.1 reader, the same-reader comparison reverses (0.5672 AKCR at 24K vs 0.4311 oracle at 78K), but we treat this as diagnostic of long-context degradation in GLM-5.1 (consistent with lost-in-the-middle behaviour, Liu et al. 2024), not as evidence that AKCR exceeds full context under a well-behaved reader.

Block B: same-reader retrieval vs. chunk-cosine retrieval. At the same $b = 24$K budget, the chunk-cosine retrieval baseline (DOS+chunks: rank chunks by $\cos(q, c)$, pack into the budget) under GLM-4.7 scores 0.5227, below the AKCR same-reader score of 0.5543 (within run-to-run variance characterised in §3.4). Under GLM-5.1, an AKCR

v1 multi-run sample (three full reruns plus one partial rerun, mean 0.5552 ± 0.053 across observed accuracies) is consistent with, but not strictly above, the single-run DOS+chunks number (0.5389). We therefore claim a same-reader retrieval improvement at GLM-4.7, and a same-reader retrieval result that is above or at parity with chunk-cosine at GLM-5.1 within noise; we do not overclaim a robust same-reader win at GLM-5.1.

Block C: cross-reader deployment observation. With Claude Opus 4.7 as retrieval reader, HACS-AKCR at $b = 24$K reaches 0.6721 ± 0.007 over three reruns. The GLM-4.7 long-context oracle on the full $\sim$78K-char book scores 0.6450. We report this only as a deployment-style data point: a stronger reader on $\sim$30% of the book producing comparable judge accuracy to a weaker reader on the full book. It is not evidence about retrieval vs. full context under a matched reader, and we do not treat it as a primary scientific claim. We further caution that this observation depends on the relative reading capability of Opus 4.7 vs. GLM-4.7 on this benchmark; with a different reader pair, the direction can plausibly invert.

Pareto across budgets. On a single-run budget sweep ($b \in \{2, 4, 8, 12, 16, 24\}$K chars, GLM-5.1 reader), AKCR v1 was above DOS+chunks at five of six budgets, with the largest gap $+8.3$pp at $b = 12$K and a tie at $b = 8$K. The associated single-run paired-bootstrap P-values look strong, but because the underlying run-to-run dispersion (§3.4) is comparable to a few of these gaps we treat the qualitative ordering, not the per-budget significance, as the take-away.

Cost and scalability note. DEC-PROP extraction is a one-time corpus-side cost ($\sim$¥150 for $4{,}929$ chunks at our concurrency setting); it is not on the per-query path. BGE-M3 encoding runs on Apple MPS; per-query retrieval over one book is a dense atom-query matmul plus an $O(K)$ source-map collapse. A many-book deployment would need sharding or ANN routing before the atom step. We did not evaluate the harder setting where the retriever must search a 100-book distractor pool; in that setting, document-level routing errors and cross-document entity distractors could make propagation more useful than it is in our source-restricted experiments.

3.3. Scoring-rule ablation

The atom-keyed retrieval admits a family of chunk-scoring rules. Define the top-K atom set T_q for query q and let $s(t) = \cos(q, t)$ be the dense atom-query score. We compare: (i) **v1**, atom-best: rank chunks by their highest-scoring atom in T_q; (ii) **MAR-sum**: chunk score $= \sum_{t \in c, t \in T_q} s(t)$; (iii) **MAR-lse** ($\tau = 5$): log-sum-exp of in-chunk top-K atom scores; (iv) **HACS** (α): $\alpha \cdot \max_{t \in c} s(t) + (1-\alpha) \cdot \cos(q, c)$; (v) **ACRP**: same chunks as v1, but the top-15 atoms are also prepended to the reader prompt as "key facts." Table 2 reports paired-bootstrap deltas vs a fresh v1 rerun ($n = 1{,}169$, GLM-5.1 reader, $b = 24{,}000$).

Reading the ablation. In our setting, HACS $\alpha = 0.8$ ties v1 (within run-to-run noise; §3.4); HACS $\alpha = 0.7$, MAR-sum, and MAR-lse all degrade weakly. ACRP regresses sharply: prepending atoms to the reader prompt appears to induce an atoms-as-content failure mode (the reader tends to paraphrase the atom list rather than synthesise from the chunks). Among the rules we tried, atom-best max-pooling is the only reduction that consistently survives. We do not claim it is uniquely optimal in general; it is the rule we keep recommending in this single-document setup until counter-evidence appears. HACS $\alpha = 0.8$ is also the variant that drives the cross-reader headline in Table 1: the 20% chunk-cosine bonus appears to recover scene-/discourse-level context without breaking the atom-dominated ranking.

3.4. Robustness and variance-aware interpretation

Across all numbers in Table 1 we apply the following interpretation rules.

(1) Run-to-run variance. A four-run AKCR v1 sample ($b = 24$K, `temperature=0`, GLM-5.1 reader; three full runs with $n = 1{,}169$ and one partial run with $n = 824$) gave judge_acc $\in \{0.5997, 0.5595, 0.5825, 0.4790\}$, mean 0.5552 ± 0.053, range 0.121. Single-run paired-bootstrap confidence intervals are tighter than this dispersion, so we report multi-run means where feasible and treat sub-3pp single-run gaps as inconclusive.

(2) Comparison to chunk baseline across budgets. On the GLM-5.1 budget sweep, AKCR v1 was above DOS+chunks at five of six budgets. Several of those gaps are within run-to-run noise; we therefore make a qualitative claim (AKCR is at least at parity with chunk-cosine across budgets), not a strong significance claim per budget.

(3) Oracle-gap retention. Defining $\text{retained_oracle_fraction} = \text{AKCR}_{24\text{K}}/\text{Oracle}_{78\text{K}}$ under a matched reader, we obtain $\sim 0.5543/0.6450 \approx 0.86$ for GLM-4.7 and $\sim 0.7338/0.8322 \approx 0.88$ for Opus on the $n = 149$ subset, i.e., AKCR retains roughly 86%–88% of full-context accuracy at $\sim$30% of input chars. Defining $\text{gap_closed_vs_chunk} = (\text{AKCR}_{24\text{K}} - \text{Chunk}_{24\text{K}})/(\text{Oracle}_{78\text{K}} - \text{Chunk}_{24\text{K}})$, we obtain at GLM-4.7 roughly $(0.5543 - 0.5227)/(0.6450 - 0.5227) \approx 0.26$, i.e., AKCR at the matched 24K budget closes about a quarter of the gap between chunk-cosine retrieval and the full-context oracle. Both ratios are budget- and reader-dependent and we report them as descriptive metrics, not as headline numbers.

Table 1. Empirical results on full-book NarrativeQA, $b = 24{,}000$ chars, $n = 1{,}169$ unless noted. BLOCK A reports same-reader retrieval vs. full-context oracle; BLOCK B reports same-reader AKCR vs. chunk-cosine retrieval at the same budget; BLOCK C is the cross-reader deployment note (§3.2). Same-reader AKCR remains below the same-reader oracle when the oracle is reliable (GLM-4.7, Opus); the GLM-5.1 oracle row is marked as diagnostic since it is degraded by long-context behaviour at 78K chars. "±" is the standard deviation across reruns where reported.

Method	Reader	judge_acc	gap to same-reader oracle
BLOCK A: SAME-READER RETRIEVAL VS. FULL-CONTEXT ORACLE			
Long-context oracle (78K)	GLM-4.7	0.6450	—
HACS-AKCR ($\alpha = 0.8$, 24K)	GLM-4.7	0.5543	−9.07pp
Long-context oracle (78K, $n = 149$ subset)	Opus 4.7	0.8322	—
HACS-AKCR (24K, $n = 149$ subset)	Opus 4.7	0.7338	−9.84pp
Long-context oracle (78K) *(diagnostic)*	GLM-5.1	0.4311	—
AKCR v1 (24K, 4-run sample) ‡	GLM-5.1	0.5552 ± 0.053	+12.4pp †
BLOCK B: SAME-READER RETRIEVAL VS. CHUNK-COSINE BASELINE			
DOS+chunks (chunk-cosine, 24K)	GLM-4.7	0.5227	—
HACS-AKCR ($\alpha = 0.8$, 24K)	GLM-4.7	0.5543	—
DOS+chunks (24K)	GLM-5.1	0.5389	—
AKCR v1 (24K, 4-run sample) ‡	GLM-5.1	0.5552 ± 0.053	—
Atoms-as-content (GAR V013, 24K)	GLM-5.1	0.4594	—
BLOCK C: CROSS-READER DEPLOYMENT OBSERVATION			
HACS-AKCR (24K, 3 reruns)	Opus 4.7	0.6721 ± 0.007	—
Long-context oracle (78K)	GLM-4.7	0.6450	—

† *Diagnostic only*: we read AKCR (24K) above the GLM-5.1 full-context oracle (78K) as a reflection of long-context degradation in GLM-5.1, not as a retrieval-vs-context win. ‡ Four-run sample: three full runs ($n = 1{,}169$) and one partial run ($n = 824$); mean and SD are across the four observed accuracies.

Table 2. Scoring-rule ablation: paired bootstrap vs v1 rerun #2 (May 8; $n = 1{,}169$, GLM-5.1 reader). HACS $\alpha = 0.8$ ties v1 within noise; alternatives degrade or tie. ACRP regresses (−7.3pp) because atoms-as-pointer pulls the reader toward atoms-as-content failure.

Rule	judge_acc	Δpp	95% CI	$P(> v1)$
v1 ($\alpha = 1.0$)	0.5595	—	—	—
HACS $\alpha = 0.8$	0.5672	+0.77	$[-1.71, +3.25]$	0.72
HACS $\alpha = 0.7$	0.5423	−1.71	$[-4.28, +0.77]$	0.08
MAR-sum	0.5458	−1.37	$[-3.93, +1.11]$	0.13
MAR-lse $\tau = 5$	0.5355	−2.40	$[-4.96, +0.17]$	0.03
ACRP $K_{\mathrm{kf}} = 15$	0.4867	−7.27	$[-10.0, -4.62]$	0.000

(4) GLM-5.1 oracle is diagnostic. We mark the GLM-5.1 full-context oracle row in Table 1 as diagnostic: at 78K-char inputs the GLM-5.1 oracle is degraded relative to GLM-4.7 in a manner consistent with long-context behaviour reported in the literature (Liu et al., 2024). Same-reader AKCR exceeding the GLM-5.1 oracle should not be read as a general retrieval-vs-context result.

(5) Negative effect sizes. The two negatives in §4 have effect sizes ($|\Delta| \geq 2.4$pp on the relevant comparisons) larger than the per-run dispersion above, and the LearnRet $\mathrm{ppr}_\alpha \to 0.95$ result is structural rather than a bootstrap effect.

4. Lessons from Negatives

We report two negative results that together bound the position. Both are negative outcomes for design choices a graph-foundation pipeline might try first; we read them as constraints on *when* graph-side propagation or graph-side payloads add value, not as evidence against propagation in general.

4.1. Learned propagation self-disables on saturated single-book seeds

We built a typed multi-edge graph over atoms (entity overlap, shared subject, scene index, and edge-hint relations) and ran Personalized PageRank with hand-tuned weights. Hand-tuned PPR lost 3.7pp F1@15 vs. dense-only on the same retrieval target. We then made all 16 graph hyperparameters trainable as `nn.Parameter`, used the APPNP closed-form solve (Klicpera et al., 2019), and trained with a multi-positive InfoNCE loss (LearnRet). The optimizer drove $\mathrm{ppr}_\alpha \to 0.95 \pm 0.01$, which makes the propagation update reduce, in the limit, to the dense seed. A natural interpretation is that the dense atom seed is already sufficient when the candidate pool is restricted to a single book of $\sim$5,000 chunks: the structured graph adds little that the seed has not already captured. We do not claim propagation is useless in general; multi-hop encyclopedic QA, multi-document settings, and schema-grounded KGQA may differ. We also cannot separate "propagation is unhelpful" from "these edges are the wrong edges": manually verified temporal/causal links or a graph with cross-document bridge nodes may change the result. We claim only that, in this single-book regime with this edge typology, the learned

propagator did not find a non-trivial use of the graph.

4.2. Atoms-as-content caps at 0.46 regardless of format

We tested four atom-feed-to-reader variants: flat bullet list of V010 atoms, 0.307 at GLM-4.7; GAR V010 (graph format with edges), 0.458 at GLM-5.1; GAR V012 (richer extraction with verbatim sources and typed edges), 0.458; GAR V013 (verbatim source as displayed text plus modality annotations), 0.459. None breaks the 0.46 ceiling. Stronger extractors do not change the ceiling: V010 (GLM-4.7 extractor, mean lexical recall 0.859), V011 (GLM-5.1 extractor, 0.895), V012 (GLM-5.1 with verbatim+edges+modality, more selective). The bottleneck is not extraction quality; it is that decontextualized propositions lose the narrative voice the reader needs to compose answers. Hybrid retrieval (atoms + chunks + summaries) at $b = 24{,}000$ scores 0.519; adding RAPTOR-style cluster summaries does not exceed 0.431 at $b = 16{,}000$. We do not count this as a full RAPTOR reproduction, since it does not reimplement RAPTOR's complete indexing and retrieval stack; it is a stress test of whether summary payloads fix the atoms-as-content failure. None of these variants break 0.6.

5. Discussion

One-paragraph summary. In our setup, once the graph token is fine-grained enough that a single dense atom seed concentrates probability mass on the answer region, a learned propagator over typed atom edges did not find a non-trivial use of the graph, and feeding atoms directly to the reader hurt rather than helped. We read this as evidence that, in this regime, the choice of graph token is worth studying before adding a propagation operator – not as a claim that propagation is unhelpful in general.

Where this position is bounded. Our results are restricted to long-document QA over narrative and semi-narrative text in NarrativeQA. We do not claim the position generalises to multi-hop encyclopedic QA (e.g., HotpotQA, MuSiQue), where edge traversal can connect facts no single chunk contains; to schema-grounded KGQA; or to non-narrative long documents (legal, scientific, code). Each is a candidate stress test and a candidate falsifier. In particular, distractor-heavy multi-hop benchmarks such as HotpotQA are a natural future stress test for AKCR: when distractor paragraphs are crafted to share entities with the query, a max-pool over atom cosines is no longer obviously the right ranking signal, since high-similarity atoms may come from distractors. We frame that as a hypothesis to be tested, not as a reported result of this paper.

Baselines not yet settled. A fair head-to-head with HippoRAG, HippoRAG 2, GraphRAG, or full RAPTOR is still missing. Those systems bring their own extraction prompts, graph construction choices, retrieval hyperparameters, and target regimes; running them naively on our 40-book NarrativeQA slice would risk comparing implementations rather than interfaces. The claims in this paper therefore rest on matched-reader oracle gaps, matched-budget chunk-cosine baselines, scoring-rule ablations, and negative controls, not on a comprehensive graph-RAG leaderboard.

Open questions for the GFM community. (i) *What is the right tokenization granularity, and how should it scale with document length?* Atomic propositions work in our setup; sub-atomic fragments may overshoot. A document-length-dependent "atoms-per-chunk" study, with consistent extraction quality, would be useful. (ii) *When does graph propagation add value on top of a strong dense atomic seed?* Our learned propagator self-disabling is a falsification signal, not a general refutation; multi-hop, multi-document, or sparse-coverage regimes are the natural places to look for the boundary.

6. Limitations

We evaluate on a single benchmark family (NarrativeQA full-book) and three commercial/API readers (GLM-4.7, GLM-5.1, Claude Opus 4.7); the generalisation in §5 remains a hypothesis until tested on multi-hop QA, schema-grounded KGQA, non-narrative long documents, and multi-document distractor pools. Our negative result on learned propagation is specific to a single-book candidate pool, this particular edge typology, and our APPNP-style parameterisation; a different graph construction or a multi-document pool may reach a different conclusion. The DEC-PROP extractor is itself an LLM and inherits LLM extraction artifacts; we mitigate with a coverage-and-faithfulness critic pass but cannot guarantee zero hallucinated or missed atoms. The judge is GLM-4.7; we use a shared judge across all conditions to control for judge bias, but the absolute number 0.6450 is judge-conditional and should not be compared directly to NarrativeQA scores in prior work using a different judge. Finally, our reader pool is predominantly commercial-API; a similar evaluation with open-weight readers (e.g., Llama or Qwen-class models) and a wider variance budget is a necessary check we have not yet done.

Declaration on LLM Use

LLMs are used as pipeline components: **GLM-4.7** for DEC-PROP atomic extraction (3 passes), as one of three reader configurations, and as the shared pairwise reference-grounded judge across all conditions; **GLM-5.1** as a second reader configuration and as an alternative extractor in ablation variants (not as judge); **Claude Opus 4.7** as the third reader configuration. All three are commercial/API models

accessed during the experimental period. The authors used automated tools for copy-editing and LaTeX formatting assistance. All experiments, numerical results, tables, claims, and interpretations were verified by the authors and remain their responsibility.

References

Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., and Liu, Z. BGE M3-Embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

Choi, E., Palomaki, J., Lamm, M., Kwiatkowski, T., Das, D., and Collins, M. Decontextualization: Making sentences stand-alone. *Transactions of the Association for Computational Linguistics (TACL)*, 9:447–461, 2021.

Davidson, D. The logical form of action sentences. In Rescher, N. (ed.), *The Logic of Decision and Action*. University of Pittsburgh Press, 1967.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., and Larson, J. From local to global: A GraphRAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., and Su, Y. HippoRAG: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jiménez Gutiérrez, B., Shu, Y., Qi, W., Zhou, S., and Su, Y. From RAG to memory: Non-parametric continual learning for large language models. *arXiv preprint arXiv:2502.14802*, 2025.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations (ICLR)*, 2019.

Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K. M., Melis, G., and Grefenstette, E. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics (TACL)*, 6:317–328, 2018.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics (TACL)*, 12: 157–173, 2024.

Perozzi, B., Fatemi, B., Zelle, D., Tsitsulin, A., Kazemi, M., Al-Rfou, R., and Halcrow, J. Let your graph do the talking: Encoding structured data for LLMs. *arXiv preprint arXiv:2402.05862*, 2024.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Sarthi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., and Manning, C. D. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *International Conference on Learning Representations (ICLR)*, 2024.