How Much Structure Should Agentic Graph Memory Build for Text Retrieval?

Satyananda Kashyap
IBM
San Jose, CA, USA
satyananda.kashyap@ibm.com

Niharika D'Souza
IBM
San Jose, CA, USA
niharika.dsouza@ibm.com

Nandana Mihindukulasooriya
IBM
New York City, NY, USA
nandana@ibm.com

Horst Samulowitz
IBM
Yorktown Heights, NY, USA
samulowitz@us.ibm.com

ABSTRACT

As agentic systems run longer and reason over larger evidence collections, simple context-window memory becomes insufficient. Knowledge-graph (KG) memory is a natural response, but existing KG-for-retrieval systems are difficult to compare because they vary construction assumptions, retrievers, reader models, metrics, and cost accounting. This paper studies KG construction as a *structure-boundary* problem, i.e., how much structure should be materialized before retrieval, and how much reasoning should be deferred to retrieval-time traversal, embeddings, and the reader LLM?

We evaluate this question in a normalized multi-hop retrieval setting across a spectrum of construction choices: heavy event/schema-rich KG construction, lightweight entity-centric construction, prompt / model-optimized extraction, Wikidata-aligned construction, token-free graph construction, and flat compiled-wiki/context baselines. The main result is that both extremes fail in different ways. Heavy construction can spend most of the token budget on abstractions that the downstream retrieval stack does not use, while graph-free or nearly graph-free baselines lose reliable multi-hop evidence composition. In our validated setting, an entity-centric construction with prompt/model optimization improves answer quality over the full high-structure baseline while reducing construction cost by more than an order of magnitude; with a stronger reader, the same lightweight graph supports further gains.

This work's contribution is therefore not a new universal KG algorithm, but an empirical recipe for setting the structure boundary: compare against both low- and high-structure baselines, keep retrieval and reader stacks fixed when testing construction choices, report quality–cost frontiers, and require each added layer of graph structure to improve downstream utility.

VLDB Workshop Reference Format:
Satyananda Kashyap, Niharika D'Souza, Nandana Mihindukulasooriya, and Horst Samulowitz. How Much Structure Should Agentic Graph Memory Build for Text Retrieval. VLDB 2026 Workshop: Agent+Graph.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

It has become increasingly clear that the fidelity of agentic systems go beyond just careful prompt curation. In fact, they are as much dependent on memory architecture as they are on memory management. A single context window may be too small, transient, and/or poorly organized to support long-running tasks over ever evolving data. When relevant evidence is missing, stale, or buried under irrelevant retrieved snippets, agents are known to hallucinate, repeat work, follow obsolete facts, and/or reason over the wrong evidence. These failures are often collectively referred to as context rot– which is the systematic degradation that occurs when an agent's working context grows without reliable mechanisms for selecting, pruning, updating, discerning, and grounding the information that matters.

A natural response is to move from ephemeral prompt context to persistent external memory. Retrieval-augmented generation (RAG) follows these lines, but flat retrieval over text chunks gives the agent little structure beyond semantic similarity. For multi-hop tasks and/or long-horizon workflows, the agent often needs to simultaneously connect entities, follow relations, preserve provenance, and distinguish stable facts from episodic mentions. This makes graph memory appealing: as it offers a natural mechanism for organising memory around entities, relations, passages, and evidence links rather than treating memory as an unordered and disjoint collection of chunks.

However, graph memory is known to introduce a new set of systems design problems. Mainly, with a graph, a sliding scale now governs how much structure can be built before retrieval. At one extreme, high-structure systems materialize rich events, schemas, summaries, communities, and typed relations before any query is asked. Examples include GraphRAG, AutoSchemaKG, KAG, and KGGen [6, 9, 19, 21]. At the other extreme end, systems may rely on flat text retrieval, long-context reading, or weakly structured summaries [22, 24]. Between these extrema lie reduced graph-memory systems; these offer the allure of retaining entities, passages, lightweight relations, or traversal structures while avoiding expensive schema/event induction [11, 13, 29, 32]. This gives rise to key questions that go beyond whether graph memory is merely useful: how much graph structure should be built before retrieval, and what reasoning can be deferred to retrieval-time traversal, embeddings, and the LLM readers.

We explain why this question is nuanced and difficult to answer by merely surveying existing literature. Mainly, in each of these paper settings, systems are free to choose their own corpi, construction models, retriever designs, reader LLMs, evaluation metrics, and cost accounting. As a result, reported accuracy numbers are often not directly comparable. Moreover, they do not reveal whether gains arise from uncovering better/more detailed graph structures, stronger readers, more/less powerful embedding models, increased construction token budgets, and/or dataset-specific choices. For agentic memory systems, this is a systems problem: graph memory is suspected to be useful, but its efficacy can only be rigorously established by treating system design as a Pareto optimization problem rather than maximising the complexity of the constructed graph structure itself. In short, a robust memory architecture should credibly justify every unit of pre-retrieval structure that it decides to build.

Our hypothesis is that graph memory has a minimum viable structure, whereby adding increasingly complex structural scaffolding is not monotonically better in terms of the downstream task. We posit that some structure is necessary, i.e. fully flat retrieval or long-context dumping loses the ability to compose evidence reliably across hops. At the same time, going much further beyond lightweight entity, relation, and passage structure, additional schema/event machinery add both cost as well as noise, significantly weighing down retrieval quality. We demonstrate that further reasoning on this minimal-viable structure may be better handled by a purposeful design of improved retrieval strategies, embedding models, and capable reader LLMs. Combined, these supersede the performance gains obtained by strictly more detailed graph structures. We test this hypothesis by evaluating graph-memory construction choices across challenging multi-hop question-answering workloads, with explicit auditing for token cost and retrieval quality–which we refer to as a cost Pareto-frontier Analysis. Concretely, this paper makes the following contributions:

(1) We frame graph-memory construction as a *structure-boundary* problem: deciding how much structure to materialize before retrieval versus how much to defer to retrieval-time traversal, embeddings, and the reader model.
(2) We run a normalized empirical study of key graph-memory design points spanning high-structure KG construction, entity-centric graph construction, grounded Wikidata-style grounding, flat retrieval, long-context reading, and LLM-index construction baselines on challenging multi-hop QA workloads.
(3) We evaluate these systems by quality–cost tradeoffs in addition to accuracy, tracking answer quality, retrieval quality, and LLM token budgets to systematically compare graph-memory construction choices on Pareto frontiers.
(4) We find evidence for a minimum viable graph structure: flat or no-graph memory is insufficient for harder multi-hop composition, but richer event/schema/grounded graph construction does not automatically improve quality. In our validated multi-hop QA setting, an entity-centric graph with prompt/model optimization reaches a stronger Pareto point than substantially more complex attributed graph constructions.

2 RELATED WORK: STRUCTURE BOUNDARY IN GRAPH MEMORY

We organize related work around the same structure-boundary question that motivates this paper: *what should be built before retrieval, what should be deferred to retrieval-time search or traversal, and what should be left to the reader LLM?*

Existing graph-memory systems occupy different points on this boundary. Some materialize rich summaries, schemas, events, and typed relations before any query is known. Others keep only lightweight entity–passage structure and rely on graph traversal, embeddings, or the reader to compose evidence at query time. A third class asks whether explicit graph structure can be avoided altogether through hierarchical summaries, long-context reading, or flat retrieval.

Maximal construction methods. GraphRAG, AutoSchemaKG, KAG, and KGGen represent the high-structure end of the design space [6, 9, 19, 21]. These systems invest substantial construction effort before retrieval: GraphRAG builds entity graphs and community summaries; AutoSchemaKG extracts entity/event triples and then induces schema-level abstractions; KAG emphasizes domain-aligned logical structure; and KGGen targets explicit knowledge-graph generation from text. The motivation is coverage: if the graph already contains the relevant entities, events, schemas, and relations, retrieval can operate over a semantically richer memory than raw chunks. The risk is that this shifts cost and error upstream. Construction-time LLM calls may add noisy abstractions, duplicate or spurious relations, and expensive structure that a downstream retriever or reader may not actually use. AutoSchemaKG is therefore our central high-structure reference point: its own MuSiQue results report nearly identical Entity-KG and Full-KG quality, already suggesting that event and schema induction may not always provide the upside necessary to justify their cost.

Flat and weakly structured memory. At the opposite extreme, systems such as RAPTOR, MemoRAG, and the LLM-as-wiki framing ask whether explicit graph memory can be replaced by text, summaries, or memory-like retrieval clues [16, 22, 24]. RAPTOR recursively summarizes text into a tree, MemoRAG uses memory cues to guide retrieval, and wiki/context baselines test whether a reader can synthesize answers from retrieved or injected textual evidence without an explicit graph. These methods reduce or eliminate graph-construction cost, but they push more burden onto retrieval ranking and the reader model. For multi-hop workloads, this is a critical low-structure baseline: if flat retrieval or long-context reading is sufficient, then graph memory is unnecessary. If it fails on harder compositional cases, then some explicit structure is doing useful work.

Reduced graph memory. A balanced middle ground would be to keep graph structure as an intermediary step but avoid maximal schema/event construction. For example, HippoRAG and HippoRAG 2 connect entities and passages and use graph-based retrieval, using personalized PageRank-style propagation to recover multi-hop evidence [12, 13]. Similarly, LightRAG and PathRAG propose lightweight graph construction and retrieval over local/global graph context paths [7, 11]. TERAG is especially relevant because

it explicitly optimizes for token-efficient graph construction, providing a better tradeoff between quality and token cost [29]. LinearRAG and iText2KG further motivate relation-light or evidence-span-centric graph construction as cheaper alternatives to fully typed KG induction [17, 32]. These directions resonate the strongest with our empirical findings for the evaluated multi-hop QA setting: *lightweight entity-centric memory often dominate richer construction on the quality–cost frontier.*

Retrieval-time traversal and agentic graph search. Another family shifts more reasoning from construction time to query time. Think-on-Graph, GraphReader, and GraphSearch use graph traversal, exploration, or agentic search procedures to decide which parts of an existing graph are relevant to a question [18, 26, 30]. These methods do not remove the need for a graph, but they draw the boundary differently: rather than materializing every useful abstraction before retrieval, they spend computation during retrieval to follow paths, expand neighborhoods, or iteratively search for supporting evidence. This is complementary to our study. We focus primarily on construction choices under a fixed retrieval/reader stack, but traversal-time reasoning is another way to move work across the structure boundary.

Grounded, temporal, and persistent memory. Recent works allude to a broader focus beyond benchmark QA towards persistent graph memory. For example, Wikontic grounds extracted knowledge in Wikidata-style neighborhoods, while Graphiti and DyG-RAG explore temporal context, updates, and dynamic graph memory [8, 23, 27]. Scale-conditioned memory evaluation further highlights that retrieval behavior can change as stored evidence grows [25]. These systems motivate the next frontier for graph memory: entity identity, provenance, freshness, and incremental maintenance.

From the practical standpoint, we see the systems motivation as distinct, yet complimentary to our empirical exploration and findings. Grounding and temporal maintenance are no doubt essential for deployed memory systems, but the cost-benefit tradeoff of richer grounding still needs to be carefully evaluated against simpler graph-memory tradeoffs auditing common retrieval and reader systems, and systematic token budgeting. Under this formalization and taxonomy, we can better shed light on why paper-to-paper accuracy comparisons can often be misleading. A reported gain may come from graph construction, but it may also come from a stronger embedding model, reader model, retriever, corpus choice, or token budget. Our study therefore treats prior systems as design points on a shared boundary as opposed to opaque leaderboard entries. The goal is not to declare one architecture universally best, but to identify which kinds of pre-retrieval structure remain useful after normalizing the surrounding retrieval and reader stack.

Table 1 summarizes this structure-boundary taxonomy and the role each family plays in our empirical comparison.

3 PROBLEM SETTING: DOCUMENT-TO-KG MEMORY CONSTRUCTION

We study document-to-graph memory construction for multi-hop question answering. Given a document corpus D, a construction procedure builds an external memory $C(D)$ before any particular question is asked. At inference time, a question q is used to retrieve evidence from this memory through graph traversal, vector search, lexical search, or some combination of these mechanisms. A reader LLM then consumes the retrieved evidence e_q to produces an answer a, i.e. $q \rightarrow C(D) : e_q \rightarrow a$. In this setting, graph memory is useful only if the structure built in $C(D)$ improves retrieval or answer quality enough to justify its construction cost.

The central design choice is the *structure boundary*, which controls what information should be materialized before retrieval, and what reasoning should be deferred to retrieval time or to the reader model?

A high-structure construction may include entities, events, event–event relations, schema concepts, community summaries, or typed logical relations. An entity-centric construction may retain only entities, lightweight relations, and links from graph nodes back to supporting passages. A flat construction may avoid explicit graph structure altogether and rely on chunks, BM25, vector retrieval, summaries, or long-context concatenation. A grounded construction may additionally normalize entities and relations against an external ontology or knowledge base. These choices are not merely representational; they determine when and where LLM calls, latency, error, and storage are introduced into the system.

We therefore evaluate graph-memory systems as quality–cost tradeoffs rather than as accuracy-centric methods. On the cost side, we track construction tokens spent before retrieval, reader/evaluation tokens spent during question answering, LLM call counts, and wall-clock time where receipts are available. We also record graph size and indexing artifacts qualitatively when full accounting is not yet audited. On the quality side, we measure exact match (EM), token-level F1 score, and retrieval Recall@5 / passage coverage.

3.1 Datasets

MuSiQue [28] serves as our primary benchmark because it stresses connected multi-hop reasoning over distributed passage evidence. Each question is labeled by hop count, which lets us test not only aggregate answer quality but also whether a memory structure degrades as evidence chains become longer. This makes it a useful surrogate for asking where pre-retrieval structure helps, where it is redundant, and where it becomes harmful. Nevertheless, MuSiQue is still only a benchmark surrogate; it does not capture every property of deployed agent memory.

We use 2WikiMultiHopQA [14] and HotPotQA [31] as cross-dataset checks. 2WikiMultiHopQA is constructed from Wikipedia and Wikidata and includes explicit reasoning paths, making it a natural transfer test for entity- and relation-centric graph memory. HotPotQA is a widely used Wikipedia multi-hop QA benchmark with supporting-fact annotations and a different question distribution. These datasets were chosen because they are standard multi-hop QA comparators, but we treat them as transfer evidence rather than as a full benchmark sweep: answer styles, corpus construction, and supporting-fact conventions differ across the three datasets.

4 BASELINE SETUP AND REPRODUCTION

We use an AutoSchemaKG-style [6] full construction pipeline as the high-structure anchor for the study. The pipeline has four stages:

Table 1: Representative systems grouped by how much structure is built before retrieval. The grouping is conceptual; systems are not directly comparable unless normalized to the same corpus, reader, retriever, and token accounting.

Axis	Representative methods	Pre-retrieval structure	Relevance to this paper
Maximalist construction	GraphRAG [9], AutoSchemaKG [6], KAG [19], KGGen [21]	summaries, schemas, events, relations	motivates full-KG baseline
Flat / weak structure	RAPTOR [24], MemoRAG [22], wiki/context baselines [16]	summaries or text only	tests whether graph memory can be discarded
Reduced graph memory	HippoRAG(2) [12, 13], LightRAG [11], PathRAG [7], TERAG [29], LinearRAG [32]	entities, passages, paths	closest to entity-centric Pareto result
Traversal-time reasoning	ToG [26], GraphReader [18], GraphSearch [30]	graph exists; traversal deferred	shifts compute to query time
Grounded / persistent memory	Wikontic [8], Graphiti [23], DyG-RAG [27]	grounded temporal/entity memory	motivates updates and provenance

- Stage 1 filters and chunks the source corpus into bounded passage units.
- Stage 2 uses LLM extraction to produce entity–entity triples, entity–event links, and event–event relations.
- Stage 3 induces schema or concept abstractions over the extracted nodes and relations.
- Stage 4 constructs the graph index and runs HippoRAG2-style retrieval followed by reader-based question answering. HippoRAG2-style retrieval propagates relevance through the graph using Personalized PageRank (PPR), a random-walk score biased toward question-relevant seed nodes, so entity and passage connectivity directly affect what evidence reaches the reader.

Our reproduction keeps the early construction path as close as possible to the original implementation. Stages 1–3 use the original code path, and the prompts were checked against the corresponding original functions. The main unavoidable substitution is the embedding backend: the paper reports AutoSchemaKG Full-KG with NV-Embed-v2 at $EM = 31.8$ and $F1 = 47.3$ on MuSiQue, while the same full-construction setting under our Granite embedding stack reaches $EM = 29.7$ and $F1 = 43.25$. The retrieval audit points to the embedding stack as a major contributor: Granite Recall@5 was 57.6%, compared with the 74.7% Recall@5 reported for the NV-Embed-v2 HippoRAG2/AutoSchemaKG stack. We therefore treat the absolute gap to paper-reported accuracy as partly an embedding-stack difference rather than purely a construction-method difference.

This reproduced full-construction pipeline is our high-structure baseline for the rest of the paper. Its cost is large: the 1000-question validation run consumes $113.4M$ total LLM tokens, and even the 250-question search-set version consumes $42.3M$ tokens. Most of that budget is spent before retrieval on schema induction. Stage 3 alone accounts for $91.8M$ tokens in the 1000-question run and $34.1M$ tokens in the 250-question run. This is the pressure point for the central ablation: if event extraction and schema induction dominate construction cost, they need to produce enough downstream retrieval or QA benefit to justify being built at all.

5 EXPERIMENTAL METHODOLOGY

We use a tiered evaluation protocol as a strategy to elicit a reasonable signal about incremental changes made to experimental design at a lower token budget. Tier 0 uses a 250-question MuSiQue subset and serves as an indicator for identifying obviously bad directions performing cost estimation before launching larger runs. Full validation uses all of the 1000 MuSiQue questions and is final basis for the paper's main MuSiQue claims. We additionally run cross-dataset validation on 1000-question 2WikiMultiHopQA [14] and HotpotQA [31] settings for the strongest entity-only recipe, although those results should be interpreted as evidence of cross-domain transferability rather that as a standalone complete benchmark sweeps.

For answer quality, we report exact match and token-level F1. For retrieval quality, we report Recall@5 / passage coverage when the retrieved evidence can be aligned to gold supporting passages. For cost, we report measured LLM tokens from files and stage receipts, separating construction tokens, reader/evaluation tokens, and total pipeline tokens when the artifacts support that split. We also report LLM call counts and wall-clock time from receipts where available, because practical agentic systems are constrained by latency and API throughput as well as by token count.

We evaluate each design point by its position on a quality–cost frontier. Quality is measured with exact match, token-level F1, and Recall@5 / passage coverage when gold evidence is available. Cost is measured primarily in LLM tokens, separating construction-time tokens from reader-time tokens when possible; we also track wall-clock time and LLM call counts because latency and API throughput matter for practical graph-memory systems. Results with incomplete token accounting or non-comparable evaluation paths are labeled as audit/interim.

6 RESULTS

The experiments separate three questions that are often conflated in graph-memory papers. First, does high-structure KG construction improve multi-hop QA enough to justify its cost? Second, if the graph is made cheaper, can prompt and reader optimization recover or improve quality? Third, are failures due to graph construction itself, retrieval over the graph, or the reader's ability to use

the retrieved context? Table 3 summarizes the search-set and low-structure controls that guided promotion decisions, while Table 4 reports the main 1000-question validation and audit results.

6.1 Evaluating Full KG vs Entity-Only KG construction

We first question whether a full pre-retrieval (maximal KG) construction such as the AutoSchemaKG [6] pipeline is actually needed to facilitate HippoRAG2-style multi-hop QA. Specifically, the full pipeline discovers and populates the KG with entity triples, event links, event–event relations, and schema/concept abstractions, each needing dedicated LLM calls.

In contrast, our entity-only variant keeps the entity graph and passage links, but removes event extraction and schema induction. Using LLaMA-3.1-8B-Instruct [10] for extraction/schema induction, Granite embeddings [4, 5], and a LLaMA-3.3-70B [10] reader, the full construction reaches $EM = 30.4$ and $F1 = 44.58$ with $42.3M$ measured tokens on the 250-question MuSiQue search set. The entity-only ablation is strictly more optimal achieving $EM = 35.2$ and $F1 = 49.91$ with $3.15M$ tokens (Table 3). This setup helps pinpoint that the token-heavy construction stages are not really helping the retriever/reader stack. In fact, they hurt quality by inflating retrieval context and adding more than an order of magnitude in cost.

This finding matches up well with broader evidence in the literature. AutoSchemaKG's own MuSiQue ablation shows almost no gain from the full graph over the entity graph. In Table 5, Bai et al. [6] report Entity-KG at $EM = 31.4$ and $F1 = 47.2$, compared with Full-KG at $EM = 31.8$ and $F1 = 47.3$. TERAG [29] also reaches a similar conclusion from the efficiency side: using LLaMA-3.1-8B-Instruct [10] for entity and document-level concept extraction and LLaMA-3.3-70B-Instruct [10] for answer generation, it reports MuSiQue $EM/F1 = 18.8/29.6$ with $3.02M$ tokens, compared with its AutoSchemaKG + HippoRAG baseline at $EM/F1 = 23.6/36.5$ with $16.68M$ tokens [29].

Together, these results are re-iterate the same structure-boundary lesson: for this family of multi-hop QA pipelines, more pre-retrieval graph structure is not automatically better. If the retrieval loop mainly uses entities, passage links, and graph connectivity, then schema/event induction can become quite costly really quickly and pollute the agent with noise rather than useful memory.

6.2 Evaluating Prompt and Reader Optimizations

After removing event and schema induction, we ask whether the remaining entity-centric pipeline can be improved through prompt and reader choices rather than by adding more graph structure. To facilitate prompt optimization (shown to be promising for KG construction [20]), we use GEPA, a reflective prompt-evolution method that proposes and tests prompt revisions using natural-language feedback over prior attempts [2]. GEPA is a prompt optimizer that incorporates natural language reflection to learn high-level rules from trial and error via genetic algorithms implementing reinforcement learning. Within a given Agentic AI system, GEPA can sample across distributions governing decision trajectories (e.g., reasoning, tool calls, and tool outputs), reflect on them in natural language to diagnose problems, propose and implement prompt refinements, distilling lessons across a Pareto frontier of its own attempts. In our setting, GEPA is applied to the extraction prompt: the graph structure remains entity-centric, but the prompt used to decide which entities and relations are extracted is optimized.

Prompt optimization improves the entity-only pipeline on the 250-question search set (Table 3), and the gain mostly survives on the 1000-question MuSiQue validation set (Table 4), with some expected degradation from search to validation. The validated entity-only recipe reaches an $EM = 38.1$, $F1 = 52.80$, and Recall@5 = 66.46 with roughly $13.6M$ tokens, compared with the full-construction baseline at $EM = 29.7$, $F1 = 43.25$, and $113.4M$ tokens. This is the main prompt-optimized Pareto point: better quality than the full graph at a fraction of the construction cost, without reintroducing schema induction. We find that GEPA optimal prompt does not strictly increase the size of the extracted entity-only KG, which is in line with our finding that a more dense graph does not automatically translate to improved QA performance.

Ablating over the choice of reader models can further push the performance frontier. Keeping the entity-only graph construction and HippoRAG2 retrieval path fixed, replacing the Maverick reader with Claude Sonnet 4.6 [3] raises 1000-question MuSiQue performance to $EM = 43.00, F1 = 59.68$, and Recall@5 = 69.02. This indicates that the structure boundary is controlled quite tightly by the reader model: a stronger reader can recover more from the same graph construction and retrieval paths. Taken together, both these findings have important implications for agentic systems, as it changes requirements on how much reasoning should be materialized prior to the retrieval step.

6.3 Graph Structure vs Flat Context/Wiki Retrieval

Low-structure baselines test the opposite hypothesis to automated KG construction: if explicit graph memory is unnecessary and if the reader is given enough text or retrieval clues, how well does it perform? The WikiLLM [16] framework is an alternative that introduces a deliberate simplification to graph construction. The core idea is fairly straightforward: keeping raw documents, have an LLM compile them into a wiki-like markdown knowledge base, then answer later questions by reading the wiki or indexing over it cleverly [16]. Recently, this pattern has generated widespread interest as it offers to replace expensive and cumbersome embedding indexes and graph materialization with a dynamically LLM-maintained text artifact. For MuSiQue, we experiment on this setting by compiling the 250-question corpus into 3,466 instantiated wiki articles using Llama 3.3-70B [10]. This compilation step consumed $1.33M$ ingestion tokens and produced a 336,799-word wiki, estimated at roughly $438K$ tokens.

We then tested five query-time variants, summarized in Table 3. The pure full-context variant is closest to the gist: ingest the compiled wiki into the reader context and answer directly. This naive approach failed quickly and dramatically. The wiki exceeded the $128K$ context window, requiring truncation to roughly $120K$ tokens per query; it reached only $F1 = 7.4$ on the first 80 questions, after which we abandoned this brute-force line of exploration.

As a next step, we implemented multiple retrieval variants to address practical scaling of LLM-Wiki. First, a BM25 variant retrieves the top-k compiled wiki articles per question and calls the same Llama 3.3-70B [10] reader. The best cheap wiki setting we found used $k = 10$, reaching an $F1 = 38.8$ with $518K$ reader tokens. Second, we include a no-context model-memory baseline: the reader receives only the question, with no wiki, retrieved passages, or graph context. This measures knowledge already stored in the reader's parameters, i.e. not including any additional parameterization of the wiki system. It reaches an $F1 = 25.4$ with $96K$ tokens. Third, we implement an LLM-index variant that makes the retriever itself an LLM. Here, Llama4-Scout [1] reads a roughly $108K$-token one-line article index, selects articles, and passes them to the 70B reader. It spends $31.8M$ tokens and reaches an $F1 = 38.3$, roughly matching BM25 at a far higher cost. Finally, we tested a stronger navigation variant with a categorical root `index.md`, 30 bounded shard indexes, page-level related pages and backlinks, and a two-step 70B navigator that selects shards and pages before expanding linked pages for the reader. This reaches an $EM = 26.4$, $F1 = 39.11$, still falling behind previous entity-only KG variants.

These variants make the low-structure conclusion sharper. Compiled wiki retrieval is competitive for some 2-hop questions, but BM25 performance falls to $F1 = 27.4$ on 4-hop questions and the old LLM-index variant falls further to $F1 = 13.8$. The stronger index/backlink wiki is the best wiki F1 we observed, but it improves over BM25 by only 0.27 F1 while using roughly 18x more query-time tokens and covering fewer of the gold evidence passages needed to answer the questions (47.33% vs 59.5%). Full-context reading fails because the compiled corpus does not fit comfortably in context, and the no-context model-memory baseline confirms that raw model knowledge alone is a poor substitute for actual retrieval. The broader conclusion is therefore not to "discard the graph."

These explorations in conjunction with the findings in Section 6.1 suggest that a practical yet quality alternative lies somewhere between flat retrieval and maximal schema induction: build enough graph structure to connect entities, passages, and evidence paths, but avoid construction-time abstractions that the retrieval/reader stack is known to not be able to exploit efficiently.

6.4 Wikidata-Aligned Construction and Direct retrieval Graph Context vs Linear Structure

Wikontic-style [8] construction tests yet another kind of memory structure. At a high-level, it consists of Wikidata-aligned entity normalization, ontology-aware relation refinement, and qualifiers, as opposed to induced full-scale AutoSchemaKG-style schemas and event population into memory. Crucially, it treats the local graph realization and relevant subgraph structure as a first-class object during retrieval. Thus, unlike many RAG-based methods, linking back to the original reference passage through a global KG structure is skipped. Instead, per query, an ontology-aligned subgraph from the KG is injected as context to the LLM reader.

To separate the effect of higher-quality graph construction from the effect of directly injecting KG context into the LLM reader, we again set up the tiered experimental protocol. The 250-question Wikontic search arms in Table 3 looked strong enough to promote, however, the 1000-question validation in Table 4 flips the interpretation. We validated three Wikontic-style graph builds—Granite 4.1 30B, Llama 3.3 70B, and GPT-OSS-20B-low extraction/refinement—while keeping the HippoRAG2-style retrieval path and Maverick reader fixed. They reach $F1 = 52.09$, 51.40, and 50.85, respectively, compared with $F1 = 52.80$ for the operationally simpler entity-only+GEPA recipe. The same pattern holds with a stronger reader: swapping Sonnet 4.6 into the entity-only HippoRAG2 path reaches an $F1 = 59.68$, while a Wikontic-style graph with the same HippoRAG2+Sonnet stack reaches an $F1 = 58.66$, but at a lower Recall@5. In this retrieval stack, richer Wikidata-aligned construction improves neither recall nor answer F1 enough to justify treating ontology alignment as a monotonic win.

Direct graph context provides a different diagnostic signal. Native Wikontic-style QA exposes dense triplet neighborhoods directly to the reader instead of converting the graph back into top-k passage retrieval. Because this path does not produce a ranked top-5 passage list, passage-level Recall@5 is not defined in the same way as it is for HippoRAG2 rows; we therefore leave Recall@5 blank for Native/direct Wikontic in Table 4 rather than mixing metrics. With Sonnet 4.6 and the same EM/F1 scoring used for the other QA runs, this direct graph-context variant reaches $EM = 44.4$ and $F1 = 54.48$ on MuSiQue. This is above the Maverick entity-only result but below the simpler entity-only+GEPA graph with the same Sonnet reader ($F1 = 59.68$). The takeaway is therefore limited but useful: direct graph context can be a viable reader interface, but Wikidata-aligned construction is still not a monotonic win under our normalized comparison.

On the other hand, LinearRAG [32] probes the opposite end of the structure spectrum: it builds a lightweight passage–sentence–entity graph from tokenization, NER, and embeddings, with LLM-based graph construction skipped entirely. This makes construction extremely cheap and attractive. Our corrected official-code replication matches the LinearRAG paper's own MuSiQue metric regime: with GPT-4o [15], we obtain 36.61% gold-string containment accuracy and 40–45% LLM-judge accuracy, in line with or above the paper's reported LinearRAG Contain/GPT-Acc numbers.

Under our exact-match/token-F1 scorer, however, the same predictions reach only $EM = 16.9$ and $F1 = 30.90$. The gap is mostly output-format driven: LinearRAG's official prompt often emits full-sentence answers, and containment or judge metrics reward answer-containing prose, while EM/F1 penalize verbosity relative to concise answer spans. At the same time, this result is not a strict apples-to-apples retrieval comparison because LinearRAG's released MuSiQue corpus uses the same questions but a different 1,354-chunk substrate. It is nevertheless informative for our frontier: relation-free graph construction is valid as a paper-style replication, but it is weak under the exact-span EM/F1 metric used for the rest of our MuSiQue comparisons.

Taken together, these results further narrow the landscape of viable system design for this problem. Flat wiki retrieval and LinearRAG-style relation-free graph construction are weak under our EM/F1 evaluation for reliable multi-hop evidence composition. Richer Wikidata-aligned construction, while promising, also does not automatically help when the downstream retrieval stack cannot effectively exploit it. The strongest evidence still favors a middle layer: build cheap entity-centric graph structure that actually helps connect related evidence paths during retrieval. At the same time, it

is useful to avoid overly expensive construction-time abstractions only for the retriever to effectively have to learn to ignore, and to evaluate direct graph-context readers with the same EM/F1 protocol used for the other QA runs rather than proxy metrics.

6.5 Cross-Dataset Transferability: 2WikiMultiHopQA and HotPotQA

Finally, Table 4 includes 1000-question checks on 2WikiMultiHopQA [14] and HotPotQA [31]. We use these datasets as transfer evidence rather than as the center of the paper's claim. They let us ask whether the same structure-boundary pattern appears beyond MuSiQue: cheap entity-centric construction remains strong, richer Wikidata-aligned construction can be competitive but pays a much larger construction cost, and relation-free LinearRAG-style construction remains substantially lower under our EM/F1 evaluation.

The cross-dataset rows also show why raw F1 alone is not the right selection criterion. On MuSiQue, the best raw-F1 row is already the entity-centric recipe with Sonnet ($F1 = 59.68$, $13.84M$ tokens). On 2Wiki and HotPotQA, Wikontic+HippoRAG2+Sonnet achieves the highest raw F1, but the gain over the best entity-centric row is modest relative to its cost: +3.20 F1 on 2Wiki (83.03 vs 79.83) for an extra $63.70M$ tokens, or about 8.5× total cost; and +1.97 F1 on HotPotQA (81.65 vs 79.68) for an extra $85.18M$ tokens, or about 8.0× total cost. The Maverick-to-Sonnet deltas are also small relative to the construction-cost gap, which suggests that dataset difficulty and retrieval substrate matter as much as reader choice in these transfer settings. LinearRAG sits at the opposite corner: low LLM construction cost, but substantially lower EM/F1. Thus the table should be read as a quality–cost frontier, not a raw-F1 leaderboard.

We therefore avoid over-interpreting absolute cross-dataset scores. The datasets differ in answer style, corpus construction, supporting-fact conventions, and hop composition, and some baselines use their own released corpus/index substrates. The cross-dataset rows are best read as a robustness check on the qualitative design lesson: graph-memory methods should be compared by downstream quality together with construction and reader cost, not by graph richness alone.

6.6 Search-Set Ablations and Promotion Decisions

We emphasize how our tiered evaluation protocol is a key component of our experimental design and results reporting. Table 3 collects the non-headline runs used to filter out which ideas show more promise and can be promoted to 1000-question validation. Running the latter directly is expensive when performing controlled ablations in a reduced search space (eg. hyperparameter optimization within a fixed configuration).

The 250-question search set correctly identified the main direction in several cases—entity-only construction plus prompt/model optimization—but it also overestimated several Wikontic-style variants. For example, Wikontic 250-question arms reached roughly $F1 \approx 60$, yet the corresponding 1000-question HippoRAG2 validations in Table 4 fell back to parity or sometimes underperformed the entity-only recipe. This is why we separate search-set evidence from final validated claims: Tier 0 is useful for pruning and promotion in a local search space, but insufficient for binding leaderboard claims.

Table 2: MuSiQue 1000-question performance stratified by hop count. Values are token-level F1 (%). All rows use the 1000-question MuSiQue validation set. Hop buckets contain 518 two-hop, 316 three-hop, and 166 four-hop questions.

System	Overall	2-hop	3-hop	4-hop
Full AutoSchemaKG + HippoRAG2 + 70B	43.25	48.98	41.84	28.05
Entity-only + GEPA + Maverick	52.80	59.69	52.23	32.42
Entity-only + GEPA + Sonnet	59.68	66.11	62.00	35.22
Wikontic G30 + HippoRAG2 + Maverick	52.09	62.66	45.19	32.22
Wikontic G30 + HippoRAG2 + Sonnet	58.66	70.22	51.98	35.32
Native/direct Wikontic + Sonnet	54.48	61.60	53.31	34.49
LinearRAG official + GPT-4o	30.90	34.71	26.83	26.75

Table 2 adds a second view of the MuSiQue results by hop count. The pattern is consistent with the structure-boundary hypothesis. Full AutoSchemaKG-style construction degrades sharply at 4-hop ($F1 = 28.05$), and LinearRAG's relation-free construction remains weak under EM/F1 across all hop counts. Entity-only+GEPA improves every hop bucket over the full graph, but even with Sonnet the 4-hop bucket remains much lower than 2-hop and 3-hop (35.22 vs 66.11 and 62.00), showing that deeper evidence composition remains the hardest case. Wikontic under HippoRAG2 is competitive on 2-hop but does not improve 3-hop/4-hop enough to beat the simpler entity-only recipe overall. The strict native/direct Wikontic result is also strongest on 2-hop and remains comparable to, but not clearly better than, entity-only+Sonnet on 4-hop.

7 DISCUSSION: EVALUATING DESIGN TENETS AFTER NORMALIZATION

The main lesson from the normalized experiments is that construction richness does not automatically rewarded downstream performance. We see that the full AutoSchemaKG-style pipeline materializes substantially more structure than the entity-only variant, but the additional event and schema layers do not improve MuSiQue QA in our normalized setting. The entity-only recipe instead gives a better quality–cost tradeoff: it preserves the entity and evidence connectivity needed for multi-hop retrieval while removing construction stages that dominate token usage. The point is not that minimal structure is universally optimal, but that each added layer of graph structure must be carefully justified by measurable downstream utility.

This reframes standard structure-boundary questions asked. The choice is clearly not between "graph" and "no graph." The low-structure baselines show that flat retrieval and long-context reading are weak substitutes for multi-hop evidence composition, while the high-structure baseline shows that materializing more abstractions can add cost without improving answers. The useful region is therefore workload- and stack-dependent: build structure that the retriever or reader can actually use effectively for the agentic workflow, and avoid construction-time artifacts whose value does not translate to retrieval coverage, answer quality, or cost optimality.

Table 3: Search-set ablations and low-structure controls used to motivate the validated experiments. These rows are separated from Table 4 because 250-question search results can overstate quality and low-structure controls are not apples-to-apples graph-construction validations.

System	Dataset	Scope	EM	F1	Tokens	Use in paper
Full KG Tier0	MuSiQue	250q	30.4	44.58	42.3M	calibration only
Entity-only KG	MuSiQue	250q	35.2	49.91	3.15M	ablation evidence
Entity-only + GEPA + Maverick	MuSiQue	250q	42.8	56.79	~4.6M	search-set champion; validated at 1000q
Wikontic Granite 4.1 8B	MuSiQue	250q	42.0	56.17	56.5M	search signal only
Wikontic Granite 4.1 30B	MuSiQue	250q	47.6	60.26	audit	optimistic vs 1000q
Wikontic Llama-3.3-70B graph	MuSiQue	250q	45.6	61.11	audit	optimistic vs 1000q
Wikontic GPT-OSS-20B-low graph	MuSiQue	250q	46.0	59.73	audit	optimistic vs 1000q
Compiled wiki + BM25 retrieval	MuSiQue	250q	25.6	38.8	0.518M	best cheap wiki setting; weak at 4-hop
Full compiled-wiki context	MuSiQue	80q partial	1.2	7.4	~120K/query	lost-in-the-middle; timed out
Parametric-only reader	MuSiQue	250q	15.2	25.4	0.096M	no-context counterpoint
Compiled wiki + LLM-index retrieval	MuSiQue	250q	25.2	38.3	31.8M	BM25-like F1 at much higher cost
Strong wiki index + backlinks	MuSiQue	250q	26.4	39.11	9.486M	highest wiki F1; ~18x BM25 tokens and lower support coverage

This is why we report quality–cost frontiers rather than accuracy alone. A single $F1$ number largely obscures whether a method paid for its result with a large construction cost, a more expensive query-time reasoning, a stronger stronger readers, different embeddings, or a different corpus. For graph memory, the meaningful comparison is answer quality together with construction tokens, reader tokens, retrieval quality, and validation scope.

The GEPA results reinforce the same point methodologically: search-set optimization can be useful, but headline claims require full validation because small subsets can overstate gains. Wikidata alignment should therefore be read as another construction choice, not as a guaranteed QA improvement. Our results clearly demonstrate that in this MuSiQue-centered QA retrieval setting, richer construction must earn its cost against a cheap entity-centric baseline. At the same time, we caution against overly generalizing any findings to other applications prior to rigorous experimentation.

The failure modes differ across the structure spectrum. Full AutoSchemaKG-style construction appears to over-build for HippoRAG2/Personalized Page Rank (PPR) retrieval: event and schema nodes add large construction cost, but the retrieval interface primarily benefits from entity–passage connectivity. Compiled-wiki methods under-build: they produce useful text artifacts, but retrieval covers fewer gold evidence passages and degrades sharply at higher hop counts. LinearRAG occupies a different low-construction corner: its relation-free graph is cheap, and it reproduces its paper-style containment/judge metrics, but it is weak under exact-span EM/F1. These failures support the structure-boundary view: the useful structure is one that the retriever and reader can actually exploit, not the richest structure that can be materialized.

8 IMPLICATIONS FOR GRAPH-MEMORY SYSTEM DESIGN

These results point toward graph-memory systems with explicit structure-depth controls. Users should not opt to pay for maximal graph construction as a default. A practical evaluation stack should be able to compare modes such as entity-only, entity-plus-types, event-aware, schema-aware, Wikidata-aligned, and summary or community-based construction. The right setting depends on the workload and on the retrieval/reader stack. For MuSiQue-style multi-hop QA, cheap entity-centric construction is a strong default because it captures the connectivity needed for evidence composition without paying for unused abstractions.

Construction should also be workload-aware. Offline graph completeness is not the same as downstream usefulness. If the target workload is multi-hop QA, construction should optimize retrieval coverage, answer $F1$, and evidence-path quality rather than graph aesthetic properties such as schema richness or ontology density. The system should make this boundary measurable, and facilitated by iterative design rather than hard-coding one universal graph representation.

Agentic optimization loops are promising but need guardrails. For example, GEPA-style prompt search can discover better extraction prompts, but it can also overfit to small search sets, produce brittle formats, or optimize proxy metrics that do not transfer to end-to-end QA. Such optimization should therefore be treated as an experimental loop with gates: parse validity checks, leakage checks, Tier 0/Tier 1 validation, and explicit token accounting. The goal is not autonomous exploration for its own sake, but controlled movement along a quality–cost frontier.

This is close in spirit to LinearRAG's motivation: avoid spending LLM tokens on construction that downstream retrieval does not need. Our results agree with that direction but not with the strongest token-free version of it. In our normalized MuSiQue EM/F1 evaluation, LinearRAG-style construction under-builds the memory, while full event/schema construction over-builds it. The most performant architectural compromise is therefore hybrid rather than novel for its own sake: build cheap entity/passage structure by default, retrieve through graph traversal, personalized PageRank (PPR), paths, or hybrid lexical/vector search, and leave some reasoning to the reader. The key design principle is to move work to the cheapest stage that can perform it reliably, while measuring whether each additional layer of structure improves the quality–cost frontier.

Table 4: Main 1000-question validation results with token accounting. The left column reports ΔF1 points / Δ total tokens relative to the same dataset's Entity-only + GEPA reference; matched-reader references are used where available, and rows without a matched reader use the Maverick reference. Dashes indicate not run, not applicable, or no LLM construction cost. Wikontic construction tokens use merged faithful token-audit reruns; MuSiQue uses max20 paragraphs and 2Wiki/HotPotQA use max10 paragraphs.

ΔF1 / Δ tok.	System / construction	Reader	EM	F1	R@5	Constr. tok.	Reader tok.	Total tok.
MuSiQue								
−9.55 / +102.87M	Full AutoSchemaKG-style KG	Llama-3.3-70B	29.7	43.25	57.64	113.38M	3.14M	116.52M
Ref.	Entity-only + GEPA	Maverick	38.1	52.80	66.46	11.06M	2.59M	13.65M
Ref.		Sonnet 4.6	43.0	59.68	69.02	11.06M	2.78M	13.84M
−0.71 / +171.84M	Wikontic + HippoRAG2	Maverick	37.9	52.09	62.10	181.78M	3.72M	185.49M
−1.02 / +172.03M		Sonnet 4.6	44.3	58.66	63.86	181.78M	4.09M	185.87M
−5.20 / +174.50M	Native/direct Wikontic	Sonnet 4.6	44.4	54.48	–	181.78M	6.56M	188.34M
−22.35 / −8.01M	LinearRAG official-code	GPT-4o	16.5	30.45	–	–	5.64M	5.64M
2Wiki								
Ref.	Entity-only + GEPA	Maverick	69.4	78.53	87.76	5.84M	2.51M	8.35M
Ref.		Sonnet 4.6	69.1	79.83	88.24	5.84M	2.68M	8.52M
+2.73 / +63.49M	Wikontic + HippoRAG2	Maverick	73.2	81.26	94.23	68.62M	3.22M	71.84M
+3.20 / +63.70M		Sonnet 4.6	73.7	83.03	94.18	68.62M	3.61M	72.22M
−5.38 / +63.39M	Native/direct Wikontic	Sonnet 4.6	65.3	74.45	–	68.62M	3.29M	71.91M
−23.29 / −2.64M	LinearRAG official-code	GPT-4o	34.6	55.24	–	–	5.71M	5.71M
HotPotQA								
Ref.	Entity-only + GEPA	Maverick	61.3	76.94	91.17	9.53M	2.48M	12.01M
Ref.		Sonnet 4.6	62.6	79.68	92.10	9.53M	2.60M	12.13M
+1.35 / +85.00M	Wikontic + HippoRAG2	Maverick	62.0	78.29	94.84	93.75M	3.27M	97.01M
+1.97 / +85.18M		Sonnet 4.6	63.8	81.65	95.92	93.75M	3.56M	97.31M
−5.82 / +88.87M	Native/direct Wikontic	Sonnet 4.6	57.7	73.86	–	93.75M	7.25M	101.00M
−16.23 / −6.51M	LinearRAG official-code	GPT-4o	42.0	60.71	–	–	5.50M	5.50M

9 LIMITATIONS

MuSiQue is a surrogate benchmark. It is useful because it stresses connected multi-hop reasoning over distributed evidence, but it is not a complete evaluation of graph memory. Our cross-dataset checks on 2WikiMultiHopQA and HotPotQA provide some transfer evidence, but broader workloads may unexpected shift even the most carefully optimized structure boundary. For HotPotQA, the 1000-row evaluation slice contains 944 unique sample IDs because some benchmark rows share the same underlying sample. We therefore account for construction once per unique constructed substrate rather than once per evaluation row: Wikontic construction is charged per unique sample-level graph, while Entity+GEPA constructs the deduplicated corpus chunks once. Reader/evaluation tokens are still counted over all 1000 question rows.

The frontier is also stack-dependent. Replacing the reader can shift answer quality without changing graph construction, and replacing the embedding model or retriever can change which graph structures are visible to the reader. In particular, our main comparisons use a HippoRAG2/PPR-style retrieval interface. A richer graph may be disadvantaged if the retriever primarily consumes entity–passage connectivity and does not explicitly exploit event nodes, induced schemas, Wikidata qualifiers, or ontology neighborhoods. Our results therefore evaluate richer construction under this shared retrieval/reader stack, not under every possible graph reasoning interface. Claims about graph construction should specify the reader, embedding, and retrieval stack upfront.

Search-set overfitting is another practical limitation, commonly observed across many benchmarks in typical agentic evaluations.

Several 250-question runs looked stronger than their later 1000-question validation results. This is particularly relevant for prompt optimization and model selection, where small subsets can disproportionately reward distributional prompt quirks or sample-specific extraction behavior. The paper therefore separates search-set ablations from main validation results and treats Tier 0 as a gating mechanism rather than a headline benchmark.

We report deterministic point estimates and have not run bootstrap confidence intervals or paired significance tests for our experiments. Small F1 differences, such as the roughly one-point MuSiQue gap between entity-only+Sonnet and Wikontic+Sonnet, should therefore be interpreted cautiously. Larger differences combined with order-of-magnitude token gaps are more robust as systems tradeoffs, but future versions should add bootstrap intervals over question-level predictions.

Finally, our results are best read as design evidence rather than a definitive and universal ranking of all graph-construction methods. The experiments support a practical recommendation: *compare structure choices under a shared retrieval/evaluation discipline, account for token cost, and verify that added graph layers improve downstream utility*. We expect the exact boundary to move across datasets, readers, embeddings, and retrieval stacks, but the evaluation recipe and the central caution against unmeasured construction complexity should transfer. All reported runs are backed by harness artifacts containing configurations, prompts/model identifiers where available, receipts, token ledgers, and per-question outputs.

10 CONCLUSION

This paper asks where graph-memory systems should draw the boundary between construction-time structure and reader-time reasoning. The answer from our experiments is that this boundary should be set empirically, not assumed to follow from graph richness. In a normalized setting, the useful guardrails are clear: compare against both low-structure and high-structure baselines, keep the retrieval and reader stack fixed when testing construction choices, report token cost alongside $F1$, and promote search-set wins only after full validation. Under those control knobs, the winning MuSiQue design is neither the richest graph nor the most "token-free" graph; it is the cheapest viable structure that still preserves entity and evidence connectivity required for complex and nuanced multi-hop retrieval.

Our flagship result is the entity-only, prompt-optimized recipe: with the Maverick reader, it reaches $F1 = 52.80$ on the 1000-question MuSiQue validation set with roughly $13.6M$ measured tokens, compared with the full AutoSchemaKG-style baseline at $F1 = 43.25$ and $113.4M$ tokens. With the same entity-only construction and a stronger Sonnet 4.6 reader alone, answer quality rises further to $F1 = 59.68$. This distinction matters: the construction lesson is that removing event/schema induction improves the quality–cost frontier, while the reader-swap result shows that the same lightweight graph can support substantially better answers when the reader improves.

The low- and high-structure baselines bracket the recommendations we propose. Flat wiki retrieval, full-context dumping, no-context model-memory answering, LinearRAG-style relation-free construction, and LLM-index retrieval all show that under-building loses multi-hop evidence composition under our EM/F1 evaluation. Full event/schema construction shows the opposite failure mode: extra pre-retrieval structure can add large cost without downstream gain. The recommendation is therefore not "less structure" in the abstract, but measured structure: each added layer should improve retrieval coverage, answer quality, or cost efficiency under the target reader/retriever stack.

As future work, we aim to implement this structure-boundary view for incremental KG construction over streaming or dynamic data. In this setting, full reconstruction becomes even less attractive because every update would repeatedly pay for structure that may not be used. The same guardrails should apply there: add new information around the existing graph where it improves retrieval or answer quality, preserve the cheapest useful entity/evidence connectivity, and only pay for richer schema or alignment when it moves the quality–cost frontier.

REFERENCES

[1] Aaron Adcock, Aayushi Srivastava, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pande, Abhinav Pandey, Abhinav Sharma, Abhishek Kadian, Abhishek Kumawat, Adam Kelsey, et al. 2026. The Llama 4 Herd: Architecture, Training, Evaluation, and Deployment Notes. *arXiv preprint arXiv:2601.11659* (2026).

[2] Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457* (2025).

[3] Anthropic. 2026. System Card: Claude Sonnet 4.6. Model system card. https://anthropic.com/claude-sonnet-4-6-system-card Accessed: 2026-05-30.

[4] Parul Awasthy, Aashka Trivedi, Yulong Li, Mihaela Bornea, David Cox, Abraham Daniels, Martin Franz, Gabe Goodhart, Bhavani Iyer, Vishwajeet Kumar, et al. 2025. Granite embedding models. *arXiv preprint arXiv:2502.20204* (2025).

[5] Parul Awasthy, Aashka Trivedi, Yulong Li, Meet Doshi, Riyaz Bhat, Vishwajeet Kumar, Yushu Yang, Bhavani Iyer, Abraham Daniels, Rudra Murthy, et al. 2025. Granite embedding r2 models. *arXiv preprint arXiv:2508.21085* (2025).

[6] Jiaxin Bai, Wei Fan, Qi Hu, Qing Zong, Chunyang Li, Hong Ting Tsang, Hongyu Luo, Yauwai Yim, Haoyu Huang, Xiao Zhou, et al. 2025. Autoschemakg: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. *arXiv preprint arXiv:2505.23628* (2025).

[7] Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2026. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 40. 30183–30191.

[8] Alla Chepurova, Aydar Bulatov, Mikhail Burtsev, and Yurii Kuratov. 2026. Wikontic: Constructing Wikidata-Aligned, Ontology-Aware Knowledge Graphs with Large Language Models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8304–8319.

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

[10] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[11] Zirui Guo, Lianghao Xia, Yanhua Yu, Tian Ao, and Chao Huang. 2024. Lightrag: Simple and fast retrieval-augmented generation. *arXiv preprint arXiv:2410.05779* 2, 3 (2024).

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

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

[14] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing A Multi-hop QA Dataset for Comprehensive Evaluation of Reasoning Steps. In *Proceedings of the 28th International Conference on Computational Linguistics*. 6609–6625. https://doi.org/10.18653/v1/2020.coling-main.580

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024.

Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[16] Andrej Karpathy. 2026. LLM Wiki. GitHub Gist. https://gist.github.com/karpathy/442a6bf555914893e9891c11519de94f Accessed: 2026-05-28.

[17] Yassir Lairgi, Ludovic Moncla, Rémy Cazabet, Khalid Benabdeslem, and Pierre Cléau. 2024. itext2kg: Incremental knowledge graphs construction using large language models. In *International Conference on Web Information Systems Engineering*. Springer, 214–229.

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

[19] Lei Liang, Zhongpu Bo, Zhengke Gui, Zhongshu Zhu, Ling Zhong, Peilong Zhao, Mengshu Sun, Zhiqiang Zhang, Jun Zhou, Wenguang Chen, et al. 2025. Kag: Boosting llms in professional domains via knowledge augmented generation. In *Companion Proceedings of the ACM on Web Conference 2025*. 334–343.

[20] Nandana Mihindukulasooriya, Niharika S D'Souza, Faisal Chowdhury, and Horst Samulowitz. 2025. Automatic Prompt Optimization for Knowledge Graph Construction: Insights from an Empirical Study. *arXiv preprint arXiv:2506.19773* (2025).

[21] Belinda Mo, Kyssen Yu, Joshua Kazdan, Proud Mpala, Lisa Yu, Charilaos Kanatsoulis, and Sanmi Koyejo. 2026. Kggen: Extracting knowledge graphs from plain text with language models. *Advances in Neural Information Processing Systems* 38 (2026), 30092–30115.

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

[23] Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956* (2025).

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

[25] Jiaqi Shao, Yiyi Lu, Yunzhen Zhang, and Bing Luo. 2026. When Stored Evidence Stops Being Usable: Scale-Conditioned Evaluation of Agent Memory. *arXiv preprint arXiv:2605.07313* (2026).

[26] Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *International Conference on Learning Representations*, Vol. 2024. 3868–3898.

[27] Qingyun Sun, Jiaqi Yuan, Shan He, Xiao Guan, Haonan Yuan, Xingcheng Fu, Jianxin Li, and Philip S Yu. 2025. DyG-RAG: Dynamic Graph Retrieval-Augmented Generation with Event-Centric Reasoning. *arXiv preprint arXiv:2507.13396* (2025).

[28] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multihop Questions via Single-hop Question Composition. *Transactions of the Association for Computational Linguistics* 10 (2022), 539–554.

[29] Qiao Xiao, Hong Ting Tsang, and Jiaxin Bai. 2025. TERAG: Token-Efficient Graph-Based Retrieval-Augmented Generation. *arXiv preprint arXiv:2509.18667* (2025).

[30] Cehao Yang, Xiaojun Wu, Xueyuan Lin, Chengjin Xu, Xuhui Jiang, Yuanliang Sun, Jia Li, Hui Xiong, and Jian Guo. 2025. GraphSearch: An Agentic Deep Searching Workflow for Graph Retrieval-Augmented Generation. *arXiv preprint arXiv:2509.22009* (2025).

[31] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*. 2369–2380.

[32] Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. 2025. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114* (2025).