Training a Knowledge Base: Supervised Structure Learning for Agent-Curated Document Stores

Yu Pan
University of Nebraska–Lincoln
yu.pan@unl.edu

Hongfeng Yu
University of Nebraska–Lincoln
hfyu@unl.edu

***Abstract*—Retrieval-augmented generation treats the document store as a frozen input, and the systems that instead let an agent curate one never measure what curation does to the store. We invert the framing: *the knowledge base is the model*. A training agent answers a supervised question against the current store, is shown the gold, then edits the store; an unchanged reader is later examined on a frozen snapshot under a fixed action budget. Where offline graph construction is unsupervised, (question, answer) pairs are our labels — and that supervision is what makes the structure cheap. Per point of corpus indexed it returns $1.6\times$ the action saving and $1.8\times$ the accuracy of an unsupervised entity index covering everything, using $1{,}913$ links against its $196{,}112$. On questions the store trained on, an unchanged reader spends 31% fewer actions at higher accuracy, and the result reproduces on an official PhantomWiki generation whose questions we did not write. To measure how far this reaches we introduce a *key-coverage gradient*, a probe varying how much of a question the training set touched, replacing a train/test split's pass/fail with a decay curve. Generalization proves endpoint-dependent: accuracy carries to unseen questions ($+0.167$ F1 where both of a question's keys were indexed, $+0.100$ where one was, zero where neither) while the action saving stays on trained questions. Because that decay is indexed by coverage rather than by novelty, more training extends it — and the store is undertrained, not saturated: coverage grows linearly in new questions and stops the moment training repeats them, so a hundred questions reach a quarter of the corpus and four times as many would close the gap.**

***Index Terms*—knowledge bases, LLM agents, retrieval-augmented generation, agentic RAG, memory, benchmarks**

I. Introduction

The dominant way to give language models durable knowledge is retrieval-augmented generation (RAG): embed a corpus once, retrieve at query time, never write back. A rapidly growing line of work breaks this asymmetry. Agent-maintained wikis compile sources into evolving page networks [1]–[3]; agentic memory systems accumulate and link notes across sessions [12]; and "sleep-time" agents reorganize memory between conversations [15]. The pattern has reached practice ahead of measurement: the community's own description of the wiki pattern lists a *lint* step—checking contradictions, stale claims, orphan pages, missing cross-links—as an engineering habit with no formal evaluation attached [3], and the closest published systems evaluate only downstream answer accuracy, never the store [1], [2].

We study the store directly, under the oldest framing machine learning has: *train/test*. The knowledge base is the model. Supervised questions arrive one at a time; a training agent answers each from the current store, is graded, and then, in an explicit consolidation phase, edits the store's structure so that the question's *class* becomes easier for a future reader. Evaluation freezes the store and examines an independent reader with no gold access and a fixed action budget on held-out questions.

The value produced by such training is best understood as *index construction*: a database does not change its data when an index is built, yet queries get cheaper. Each training question's verified reasoning path is materialized as *access structure*—a multi-hop chain becomes a traversable link path, a key that search handles badly gains an index document linking everything under it, so future readers pay a lookup price instead of a re-derivation price, and the one-time cost is repaid after a computable number of questions.

Contributions.

1) **Agent-trained knowledge bases, and why supervision is the efficient way to spend structure.** We let an LLM agent *train* a document store under the standard machine-learning paradigm, the knowledge base playing the role of the model: supervised (question, answer) pairs are consumed one at a time; each iteration runs a forward pass (answer from the current store) and a backward pass (with the gold revealed, *edit the store*—adding, deleting and linking documents, building index documents); held-out performance is the loss. The contrast with offline construction (GraphRAG, RAPTOR, HippoRAG) is exact in machine-learning terms: those are *unsupervised*, inferring structure from the corpus with no labels and no notion of what will be asked, while (question, answer) pairs are our labels and structure is optimized against verified answers. The advantage this buys is measurable and it is one of efficiency, not of ceiling. Per point of corpus indexed, supervised curation returns $1.6\times$ the action saving and $1.8\times$ the accuracy of an unsupervised entity index that covers everything, because it spends its structure where questions actually go: $1{,}913$ links against $196{,}112$, and 94% of ours point at a document that genuinely belongs under its key.

And the structure generalizes, on the endpoint where generalization is available to it: questions the store never saw gain $+0.167$ F1 when both of their keys were indexed during training and $+0.100$ when one was, decaying to zero when neither was. That decay is indexed by *coverage*, not by question novelty, so it recedes as training continues. Our absolute numbers trail the unsupervised baseline only because a hundred training questions reach a quarter of the corpus, and coverage grows linearly in questions consumed — the gap is training volume, not method. Supervision also composes with deployment in a way offline construction cannot: the loop consumes questions one at a time and edits in place, so a store can keep learning from live traffic and concentrate its structure on the query distribution it actually serves.

2) **A train/test protocol for knowledge bases**: two-phase (forward/backward) training iterations, single-shot gold reveal, frozen-store examination under action budgets, learning curves, and a *key-coverage gradient* — a probe that varies how much of a test question's key the training set touched, which turns "does it generalize" into a measurable decay curve rather than a yes or no.
3) **KBGym**, a contamination-free environment in the PhantomWiki style [8]: a fictional-person universe rendered as a graph of atomic single-sentence documents; ten diagnostic question classes with exact answers and support sets; deterministic, integer-exact store metrics and no LLM judge in any measurement — grading is SQuAD normalization against programmatic golds, and the one LLM judge in the system guards deduplication inside the store, never a reported number.

II. Related Work

Retrieval-augmented generation and agentic retrieval. Classic RAG retrieves from a frozen corpus with a learned dense retriever and generates conditioned on the results [16]–[19]: the corpus is an input, never an output. Agentic RAG moves retrieval into the reasoning loop, interleaving it with reasoning [9], [20], decomposing into sub-questions, retrieving on low confidence [10], or learning the retrieve-or-reason policy [11], [21]. Our reader *is* such a loop. The distinction is which side learns: that line improves the query-side policy against a fixed corpus; we hold the policy fixed and train the store. The axes are orthogonal and composable.

Building structure over a corpus. GraphRAG [4], RAPTOR [5], LightRAG [28], and HippoRAG [6], [29] build entity graphs, summary hierarchies, or PageRank-linked knowledge graphs *before* any question arrives. These are our strongest baselines and the natural contrast: their structure is unsupervised and built blind to the query distribution, ours is carved incrementally by the questions actually asked. On the parametric side, ROME and MEMIT edit factual associations inside model weights [35], [36]; we pursue the non-parametric complement — facts live in an external store that can be inspected, reorganized, and audited, with edits that transfer across models. STaR-style bootstrapping [37] keeps only verified reasoning for further training; our forward/backward iteration inherits that predict-then-learn shape, with the store rather than the weights as the learned object.

Agent-maintained stores and agent memory. Recent systems have an agent write and update wiki pages during question answering, or maintain a time-evolving wiki under a document stream [1]–[3]; STORM [27] generates Wikipedia-style articles by multi-perspective research — one-shot authorship rather than continual maintenance. A parallel line accumulates experience across episodes — external memory paging [22], linked long-term note stores [12], [23], reflection over an event stream [24], and distilled feedback or reusable workflows [7], [25], [26]. In these the artifact is a prompt or a private memory serving one agent; ours is a shared, inspectable document store whose curation quality is itself the measured outcome.

Measuring memory without contamination. The published curation systems report downstream task scores only: the store itself is never measured, there is no train/test split over questions, and the evaluation corpora are parametrically contaminated. LongMemEval and successors probe assistant memory with QA over past sessions [13]; LoCoMo evaluates very-long-term conversational memory [34]; MemDelta documents hidden confounds and argues for controlled baselines [14], supporting our no-store baseline discipline. Multi-hop QA datasets — HotpotQA [30], MuSiQue [31], 2WikiMultiHopQA [32] — supply real-text questions with annotated support, but their public corpora are contaminated for current models. PhantomWiki [8] generates contamination-free fictional universes with programmatically exact answers; our generator follows its recipe, and SQuAD-style normalization [33] provides the grading. Our protocol adds the missing measurement layer.

III. Problem Formulation

A knowledge base K is a directed graph of *documents*, each one self-contained sentence plus its outgoing edges, so that what the store learns is visible in its structure rather than hidden inside prose. Two fields are all any agent sees or writes: `text`, the one sentence `search` matches against, and `links`, the edges `read` follows — `read` returns a document together with the full text of everything it links, which is what lets a well-connected document replace a sequence of uncertain searches. An *index document* is not a distinct type but a document whose text names a key and whose worth lies in its links; nothing marks it as one, which is why an index carrying no links is indistinguishable from a sentence nobody needs. Three further fields are kept by the environment and are invisible to the agent, so that measurement does not depend on its cooperation: `origin` (which original fact a document represents, empty when authored), `flag` (untouched / authored / edited), and `absorbed` (origins folded in by a merge). Together they make the ledger exact.

The *reader* is a fixed LLM policy with `search`, `read` and `answer`, a fixed action budget M, and a memory

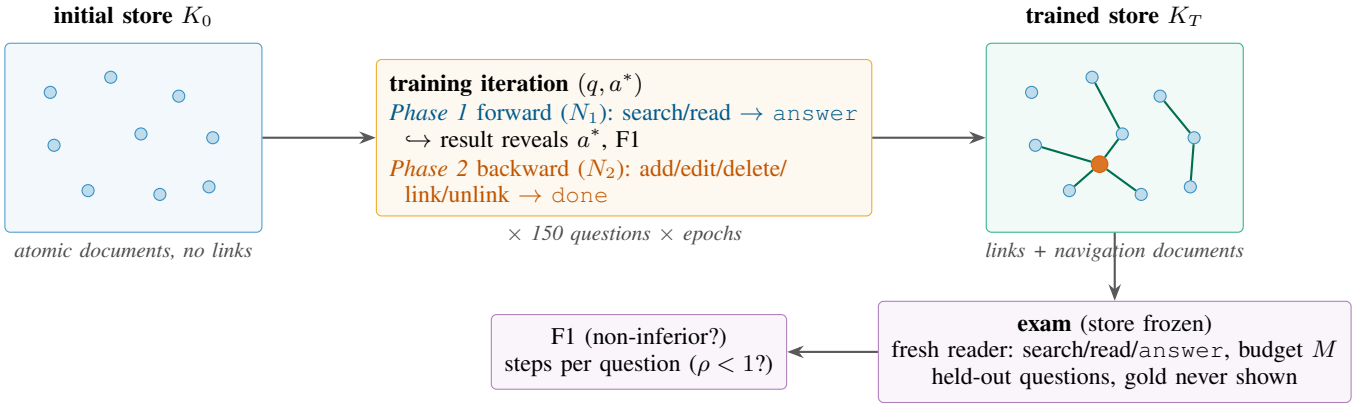

Fig. 1. The train/test protocol. A training agent consumes supervised QA pairs, answering each from the current store (gold revealed only after its single answer) and consolidating verified reasoning into structure. Evaluation freezes the store and examines an independent budgeted reader on held-out questions; the hypothesis (Eq. 1) is non-inferior accuracy at strictly lower per-question cost.

holding only the current question and the last k action–result pairs. Its behaviour on a question distribution Q defines store quality on two axes, $\mathrm{acc}(K) = \mathbb{E}_q[\mathrm{F1}(\mathrm{reader}(K, q))]$ and $\mathrm{cost}(K) = \mathbb{E}_q[\mathrm{steps}(\mathrm{reader}(K, q))]$. *Training* is any procedure consuming supervised pairs (q, a^*) and editing K, producing K_T from K_0.

What such a procedure should learn is *access structure*, and the distinction matters because the action set admits a shortcut that resembles learning. A store can gain *content* — a document stating a fact, including one the store already implies, such as an answer just verified — or *access structure*: index documents whose value is the links they carry, and edges a chain must step across. Only the second generalizes. A document recording a verified answer serves the instance that produced it, and that instance will not recur; an index over a key serves every question whose search names that key. Our hypothesis is therefore a claim about access structure, and we measure navigability alongside accuracy and cost:

$$\mathrm{acc}(K_T) \geq \mathrm{acc}(K_0) - \delta \quad \text{and} \quad \rho = \frac{\mathrm{cost}(K_T)}{\mathrm{cost}(K_0)} < 1, \quad (1)$$

with margin $\delta = 0.03$. Training amortizes: for C_{train} tokens spent and a per-question saving of $c_0 - c_T$, the break-even point

$$N^* = C_{\mathrm{train}}/(c_0 - c_T) \quad (2)$$

is the query volume after which curation has paid for itself.

The correspondence to supervised learning is exact: the store configuration is the parameter vector, one iteration is one optimization step, the training agent is the optimizer, and held-out reader performance is the loss. Because training forward passes and test readers use identical tools, budgets and the single-shot answer rule, the generalization gap is well defined — and Sec. VII-B refines it from a train/test dichotomy into a gradient over how much of a question the training set touched.

IV. The KBGym Environment

A. Universe, Store and Questions

A generator in the PhantomWiki style samples a fictional population (500 people): family trees, spouses, friendships, per-person attributes (job, hobby, city), distinct birthdates, and *name distractors* sharing first names with questioned subjects. Facts render through fixed templates into 5,864 atomic single-sentence documents ("Alice Johnson's job is arborist."), and the initial store has *zero links*: a bag of facts. Ten question categories over 26 templates. QC1–QC3 are 1–3-hop named-entity chains, natively easy for document-level retrieval, and serve as the non-inferiority floor. The rest target relations *between* documents: aggregation counts (QC4, QC8), abstention (QC5), multi-constraint joins with no name anchor (QC6), set intersection (QC7), superlatives over birthdates (QC9), reverse lookup with a uniqueness guarantee (QC10). Golds come from graph traversal, every question carries its support set, and grading is SQuAD-normalized F1. Splits are instance-disjoint: `train` 150, `test_in` 100 (unseen instances of trained templates), `test_out` 50 (one reserved template per category), `eval` 30 (drives the per-epoch curve so the test splits are touched once).

B. Actions

Table I is the complete action set. Availability is enforced by the tool schema rather than at runtime: the reader is never shown an editing tool, so it cannot decline to use one, and the curator is never shown `answer`. The reader's three actions are byte-identical in training phase 1 and at exam time, which is what makes the frozen store the only thing that differs between the two.

Two prices are set deliberately. `search` returns five documents per page and a page costs an action, so enumerating a set is possible but expensive — the twenty-nine residents of a city cost six actions, while one `read` of a complete index costs one. Closing that gap is exactly what a trained store is for, and returning sixty results at once would erase the thing being measured. Conversely `link_many` attaches

TABLE I
THE ACTION SET. *R* = AVAILABLE TO THE READER (EXAM AND TRAINING PHASE 1); *C* = AVAILABLE TO THE CURATOR (TRAINING PHASE 2). NO ACTION IS AVAILABLE TO BOTH ROLES IN A WAY THAT LETS THE READER MODIFY THE STORE.

Action	Effect	R	C
`search(q, page)`	five most similar documents; one page per action	✓	✓
`read(id)`	the document and the full text of everything it links to, one level	✓	✓
`answer(text)`	submit and end the pass	✓	
`add(text)`	new document; a near-duplicate is merged instead		✓
`edit(id, text)`	replace a document's text		✓
`delete(id)`	remove a document		✓
`link(a, b)`	one directed edge		✓
`link_many(a, T)`	up to forty edges from `a` for one action		✓
`unlink(a, b)`	remove an edge		✓
`done()`	end the curation pass		✓

up to forty targets for a single action, so building an index is cheap once its members are found: the curator's budget goes on finding, not on attaching. Batches above forty targets are rejected — the largest genuine key in this universe has thirty-six members, so a larger batch is a search result rather than a set, and truncating it silently would leave the curator believing it had built something it had not.

Provenance is tracked throughout: initial documents carry an origin, edited ones are flagged, authored ones have no origin. Coverage and duplication are therefore integer-exact and any authored content is attributable.

C. Design Rationale

Four decisions carry the methodology. *(i) No LLM judge in any measurement*: golds are exact and store metrics are integer counts over provenance, so every number here is reproducible to the digit; the one judge in the system vetoes near-duplicate writes and scores nothing. *(ii) Realistic retrieval*: `search` returns matched document text, as production vector stores do — an early variant returning only titles collapsed the reader to F1 0.0 and would have credited the trained store for repairing a crippled interface. *(iii) Atomic documents*: the store is its own chunking, so organization is expressible *only* through links and index documents, and any efficiency gain is attributable to structure. *(iv) Contamination control*: gpt-5-mini answers HotpotQA-style questions at F1 ≈ 1.0 with *no retrieval at all*, so on public corpora a reader's score conflates store with weights. A fictional universe puts the no-store score at ≈ 0, and every point of reader performance is earned through the store.

V. TRAINING PROTOCOL

Fig. 1 gives the shape of one iteration and of the exam that follows.

Phase 1 uses exactly the reader's toolset and budget, so train-forward accuracy is directly comparable to test accuracy. The single-shot `answer` reveals the gold in its result—standard supervised learning: predict, then see the label, then update. Leftover Phase-1 budget is forfeited (no smuggling forward steps into consolidation); a budget-exhausted forward scores 0 but still receives the gold, so failed questions—where repair matters most—get targeted consolidation. Every intermediate store state is reconstructible by replaying the edit trace (validated byte-exact against epoch snapshots), giving per-iteration store trajectories for analysis at no storage cost.

What the backward pass is told is deliberately uniform: no oracle localization, and no branching on how the forward pass failed. The agent receives its own trajectory, the gold answer, its F1, and one of three coarse outcomes (budget exhausted, answered wrongly, answered correctly), followed by the same instruction in every case — name the keys this question mentioned, the people, places, jobs, hobbies and relations it named, and make sure each has a complete index document: build what is missing, extend what is partial, change nothing if they are already complete.

The curation skill supplies the standing constraints rather than a procedure. An index *points*; its members belong in its links, never recited in its sentence, so the index survives the facts changing under it. One key at a time: a question naming four keys and a budget that closes one should close one, because an index of twenty-nine members is worth more than four of three, and an unreached key is picked up by the next question naming it. Precision before completeness: search returns what is similar, not what belongs, so linking a whole result set destroys the certainty an index exists to provide. Completeness is then a promise — a reader that finds an index stops searching, so a partial index does not merely underperform, it makes the reader confidently count nine where there are fourteen. And an index under which the store holds nothing is deleted rather than left standing, since it would cost a retrieval slot and return nothing. Parsimony governs all of it: duplicated documents compete in search and bury each other. That last constraint encodes the lesson of preliminary runs on an earlier environment variant, in which an agent trained *without* it quintupled an already-organized store and regressed the reader by 17 F1 points while every individual edit looked constructive.

Credit assignment needs no oracle: a wrong answer plus the revealed gold lets the agent re-search *with the answer in hand*, and what to repair follows from the difference between where it searched and where the answer turned out to live.

VI. EXPERIMENTAL DESIGN

A. Benchmarks

Two arms share one protocol and differ in who authored the questions (Table II).

Official PhantomWiki (external validity). A generator we do not control (phantom-wiki 1.0.3): 16 family trees, 405 person articles, 480 generated QA over 8 templates in four composition shapes, hop depth 1–5; two reserved templates form `test_out`. Its Prolog-derived relations carry no per-fact supports, so repair diagnostics are unavailable there, which the protocol itself does not need. A real-text arm is the obvious third benchmark and we did not run it: public corpora are

TABLE II
BENCHMARK ARMS. BOTH USE THE SAME SPLITS SCHEME, READER, ACTION SET AND BUDGETS; THE ARMS DIFFER IN WHO WROTE THE QUESTIONS.

	KBGym (main)	PhantomWiki (official)
documents	5,864 sentences	3,403 sentences from 405 articles
source	our generator	their generator
questions	330 (150/100/50/30)	330, same scheme
question author	us	PhantomWiki
supports	exact	unavailable
contamination	none	none

parametrically contaminated, so every score would need a no-store baseline that is a study in itself.

B. Baselines

All baselines are *unsupervised*: structure is induced from the corpus alone, with no access to questions or answers — the defining contrast with our trainer, for which gold answers are labels. They differ from us and from each other *only* in how the store is prepared; reader, test sets and budgets are identical. **B1, flat store**: the untrained store (= epoch 0), no structure, no build cost. **B2, GraphRAG-style** [4]: lexically extracted entities per document, co-occurrence communities, one LLM-written summary document per community linked to its members (488k build tokens). **B3, HippoRAG-2-style** [6]: (subject, relation, object) triples per document, entity-sharing links, per-entity hub documents (lexical, so ≈0 build cost). Both land their structure as documents and links *in our store format*, so the common reader can traverse it: we test their *structure* under a fixed reader rather than their retrieval algorithms, a documented adaptation — B3 forgoes Personalized PageRank.

C. Implementation

Storage and retrieval. The store is a dictionary from document id to a record of `text` and outgoing ids; there is no database, links are ids rather than copies, and a delete cascades so no edge dangles. A separate in-memory vector index (chroma, default ONNX MiniLM embeddings) serves `search` and holds nothing else: adding or editing a document marks it dirty, and the environment re-embeds dirty documents in one batch at the end of an iteration, so the index is strictly derived state. **Reproducibility.** Every action and its result is appended to a trace, so any intermediate store is rebuildable exactly by replay — without an API call, including merges, whose verdicts are recorded rather than recomputed. This is not only convenient: when an API outage killed a run 100 iterations in, the store was recovered from its trace in under two minutes.

Models and budgets. Trainer and reader are gpt-5-mini (temperature 0.3, low reasoning effort, 12k completion tokens per turn) with tool-forced decisions and static schemas; the `gpt-5-mini` alias resolved to snapshot `gpt-5-mini-2025-08-07`, which we record because the alias will move. Budgets $N_1{=}M{=}15$, $N_2{=}30$ — the backward pass is longer because it must both locate a set and attach it — and FIFO memory $K{=}30$. The main run averaged 94k tokens and 123 s per iteration (200 iterations, 18.8M tokens, 6.8 h, ≈$13), single seed.

The deduplication guard. On add or edit the store embeds the candidate, takes the three nearest documents, and consults an LLM judge when cosine similarity exceeds 0.90. The judge gets 500 completion tokens at minimal reasoning effort, which matters more than it sounds: a reasoning model spends its budget thinking before it emits anything, and at the four tokens an earlier version allowed it returned an empty string on every call, read by the caller as "not a duplicate" — the guard was wired up and inert. Measured directly, 4 and 64 tokens both yield empty output and 500 answers in about ten, so the merge counts reported here come from a judge that answered.

D. Experiment Suite

Four experiments follow, each stated with the prediction it was designed to falsify. E1 reads the training dynamics; E2, the headline, is the key-coverage gradient, whose prediction is that ρ rises monotonically from the trained group towards 1 and that the rate at which it rises *is* the transfer radius; E3 decomposes retention by question class, predicting that gains concentrate in the relation-level classes retrieval cannot solve natively; E4 audits the store itself. Break-even N^* (Eq. 2) closes the analysis.

VII. EXPERIMENT RESULTS

All results are from the final environment and the main run.

A. E1: Training Dynamics

The 500-person configuration was chosen by probing untrained stores: forward chains are natively easy at any scale (F1 1.0), while the relation-level classes leave headroom that grows with the universe (untrained F1 0.83 at 120 people against 0.67 at 500, mean steps 6.7 against 9.2), so the cost axis has roughly $3\times$ room above its $\sim$3-step floor. The main run trains on 100 questions for two epochs (200 iterations, 18.8M tokens, 6.8 h, ≈ $13). Two signals establish that the store accumulates before the frozen exams test what it is worth. Forward accuracy climbs *within* epoch 1, from 0.60 over the first fifty iterations to 0.66 over the last fifty, before any question repeats: later training questions already benefit from structure earlier ones left behind. And curation slows as the store fills — 4.9 edits per iteration in epoch 1 against 1.6 in epoch 2, 25.3 of 30 backward actions spent against 14.7, document growth 2.4 per iteration against 0.45 — because the agent increasingly finds what a question names already indexed, which is what the parsimony objective asks for. The per-epoch eval curve is flat (eval-in $0.75 \to 0.80 \to 0.80$, eval-out $1.00 \to 0.90 \to 0.90$, $n{=}20$ and 10); at that sample size it is not distinguishable from noise, and the eval split is instance-disjoint from training, which places it at the far end of the coverage gradient measured next. We report it because it was pre-registered, not because it carries weight.

B. E2: The Key-Coverage Gradient

How far does built structure reach? A train/test split answers that with a single number, which conflates two very different failures: a store that memorized its training questions and a store that generalizes but was never given enough of the corpus to cover the test set. We separate them by constructing four question groups that differ only in how much of the question the training set touched. KBGym is generated, so the full instance pool of every template is available: from it we sample, for the six two-slot trained templates, thirty questions whose *both* keys appeared in some training question, thirty with exactly one, and thirty with neither, plus thirty of the training questions themselves. All four groups are then examined on the same frozen store, by the same reader, under the same budget — only coverage varies.

Table IV is the result and Fig. 2 its training-time version. The step saving is real and it is narrow. On the questions the store trained on, the reader spends 31% fewer actions ($\rho = 0.686$, CI $[0.52, 0.84]$) at higher accuracy than on the untrained flat store. One step away — same template, both keys indexed, different question — the saving is gone ($\rho = 0.935$, CI spanning 1), and it does not return. Coverage of the keys is therefore *not* what buys the saving; having answered that exact question before is.

Two epochs, two different gains. Fig. 2a–b separates them. Across epoch 1, where each question is seen once, the trained group's accuracy moves almost all the way it is going to move ($0.633 \rightarrow 0.800$) while cost barely does ($9.20 \rightarrow 8.30$ actions). Across epoch 2, where the same hundred questions are curated a second time, cost falls the rest of the way ($8.30 \rightarrow 6.57$) and accuracy does not move at all. The second pass does not change what the store can answer; it changes how quickly. Nor does it work by completing indexes left half-built — mean out-degree is flat across the epoch (6.79 to 6.67) and the empty count barely moves (45 to 43) — it works by adding 45 more indexes aimed at the same questions. The saving we report therefore reflects two curation passes over a question, not one, which is a real qualification on the headline number and a direct argument for the online setting: a store serving repeat traffic gets more passes over the keys that matter, for free.

The transfer radius depends on which endpoint you ask about. Splitting each group by whether the flat store exhausted its budget separates two effects that the pooled ρ hides. Where B1 runs out of actions, the trained store both finishes sooner and answers better, and this persists past the exact key: F1 rises $0.30 \rightarrow 0.50$ on the trained questions, $0.18 \rightarrow 0.55$ at two keys covered, and $0.00 \rightarrow 0.60$ at one. Where B1 already finishes comfortably, the trained store is slightly *slower* off the trained set ($\rho = 1.08$ and 1.17), because its indexes occupy retrieval slots on questions that did not need them. Built structure transfers as accuracy on questions the flat store cannot finish, and as step savings only on the questions it was trained on. Reporting one number for "does it generalize" would have hidden both halves.

The same shape on questions we did not write. KBGym is our generator, so the gradient could be an artifact of templates chosen by the people who designed the method. The PhantomWiki arm answers that: its universe (3,403 documents from 405 articles) and every one of its questions come from a generator we do not control, and the protocol runs unchanged (200 iterations, 14.9M tokens). Its questions carry at most one key, so the gradient there has three points rather than four, and they align with KBGym's:

ρ (F1)	trained on	one key	neither
KBGym	0.686^* *(0.800)*	1.032 *(0.867)*	0.904^* *(0.833)*
PhantomWiki	0.769^* *(0.880)*	0.914 *(0.807)*	1.029 *(0.718)*

The trained group is the only cell significant on both benchmarks (CIs $[0.52, 0.84]$ and $[0.68, 0.86]$), and accuracy rises on both ($0.700 \rightarrow 0.800$ and $0.728 \rightarrow 0.880$). The two middle cells disagree in their ordering across benchmarks, which falsifies the monotone decay E2 was designed to test: there is no reliable partial transfer in the step endpoint, only a saving where the question was trained and none where it was not. On PhantomWiki the uncovered group is worse than neutral — F1 falls $0.765 \rightarrow 0.718$ — as indexes for other keys occupy retrieval slots that question needed. Coverage is not merely absent outside the trained set; it is mildly costly, which is the sharpest argument for extending it rather than accepting a quarter of the corpus. The full four-point gradient is unavailable on this arm because constructing one requires enumerating a template's entire instance pool and PhantomWiki ships the questions its generator produced rather than the pool behind them; the arm also carries no offline-construction baselines, so the comparison below is KBGym-only.

Coverage is the currency. B3 answers in fewer actions and at higher F1 than we do, from 100% coverage against our 27.6%: it indexes every entity in the corpus, while a hundred training questions name a quarter of it. Its ρ holds between 0.71 and 0.78 in all four groups; ours beats it where our indexes exist (0.686 against 0.777 on the trained group) and falls back to the flat store where they do not. A gradient defined by what training touched is in any case a property of our arm alone, so the comparison that treats both fairly divides each arm's gain by the coverage that produced it.

Coverage counts a source document as covered when at least one authored index links to it directly, and is reported as a share of the 5,864 originals. Three alternative readings agree, so the number is not an artifact of the definition: allowing a second hop through an index-to-index edge leaves it unchanged at 27.6% (the agent built only 100 such edges); counting only links a semantic check confirms as correct gives 27.3%; and at the level of questions rather than documents, 78 of 302 (25.8%) have their entire support set reachable from some index in one read. Both offline arms sit at 100% on every one of these readings, by construction.

Pooled over all 120 probe questions, B1 answers in 9.8 actions at F1 0.725. B3, indexing the whole corpus, reaches 7.3 actions and 0.908; our store, indexing 27.6% of it, reaches 8.7

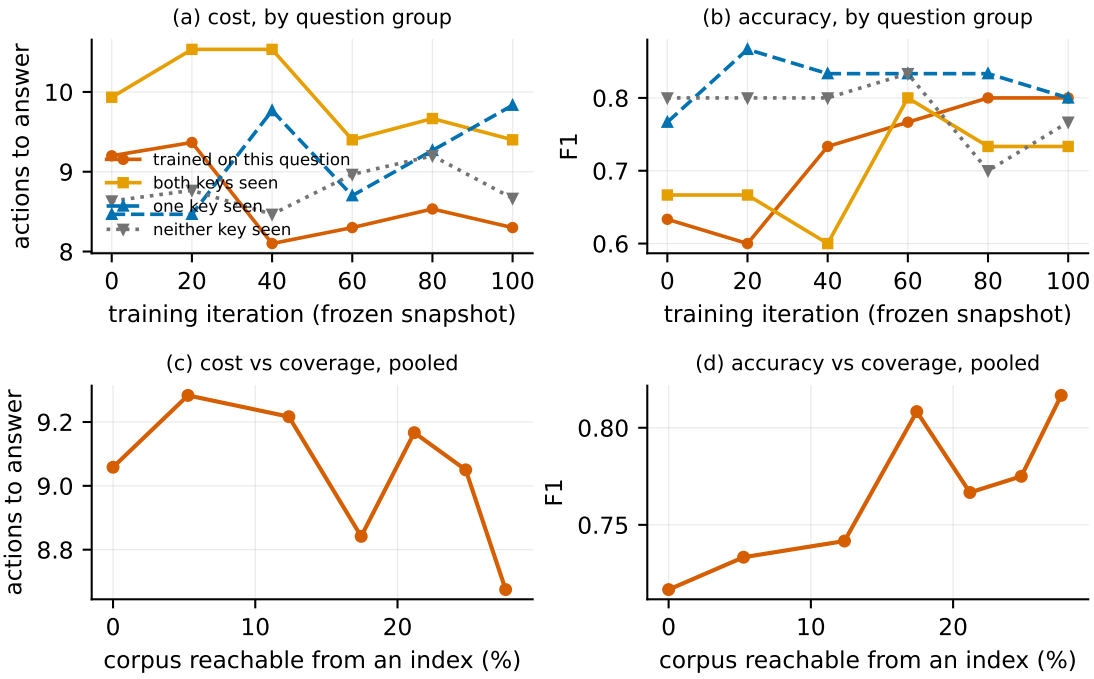

Fig. 2. Six frozen snapshots from epoch 1, read two ways. The reader, action set, budget and questions are identical at every point, so the only thing varying is the store. *(a, b)* Cost and accuracy per question group against training iteration: the trained group improves while the three untrained groups do not, and the ordering is already visible within a single epoch. *(c, d)* The same runs pooled, against the share of the corpus the store's indexes reach — the quantity training actually buys. Accuracy rises steadily with coverage; cost falls more noisily. Coverage reaches 24.8% by the end of the epoch, and the second epoch (Sec. E2) adds the remaining cost saving without adding accuracy.

TABLE III
WHAT A POINT OF CORPUS COVERAGE BUYS. EACH ARM'S GAIN OVER THE UNTRAINED FLAT STORE, POOLED OVER ALL 120 PROBE QUESTIONS, DIVIDED BY THE SHARE OF THE CORPUS ITS INDEXES REACH. THE OFFLINE ARMS INDEX EVERYTHING; OURS INDEXES THE PART ITS TRAINING QUESTIONS NAMED.

Store	coverage	actions / pt	F1 / pt
B1 flat	0%	—	—
B2 GraphRAG	100%	0.003	−0.0002
B3 HippoRAG2	100%	0.025	+0.0018
Ours (trained)	27.6%	**0.039**	**+0.0033**

and 0.817. Per point of corpus coverage that is 0.039 actions saved and +0.0033 F1 for us against 0.025 and +0.0018 for B3 — 1.6× and 1.8× more per point covered. The accuracy ratio is robust to how the probe is composed (1.7–2.2× on each group taken alone); the action ratio is not, and inverts to 0.6× if the trained group is dropped, so it is worth whatever a deployment's overlap with its training questions makes it worth. B2, which also covers everything, returns 0.003 actions and *negative* F1 per point: coverage alone is not the mechanism, and community summaries are the wrong structure regardless of how much of the corpus they span. What separates us from B3 is therefore not the quality of the structure but how much of the corpus carries any.

The store is undertrained, not saturated. This is the central qualification on every number in this paper, and Fig. 2c–d is the evidence for it. Through epoch 1 the reachable share of the corpus grows almost linearly in questions consumed, at 0.25 points per question. Epoch 2 re-asks the same hundred questions and the curve flattens immediately — 24.8% to 27.6% over a hundred further iterations. Nothing saturated; the supply of new keys ran out. Extending the epoch-1 slope reaches full coverage at roughly 400 distinct training questions and ≈ 48M tokens — about four times the training we ran, rather than a change of method. That projection is a lower bound: the largest keys are hit first, so later questions cover less each, and we state it as an extrapolation rather than a result.

The protocol is already the online one. Training consumes questions one at a time and edits in place; nothing in the loop needs the question set in advance, and the frozen-store exam is a measurement device rather than a deployment constraint. A store curated against live traffic would therefore accumulate coverage on exactly the keys its users ask about — the distribution where, by Table IV, coverage is worth the most. Offline construction cannot follow a query distribution it never sees. We did not run that experiment: separating a training phase from a frozen exam is what makes the generalization question answerable at all, and an always-learning store cannot be said to have been tested on anything. The two readings are complementary, and the online one is the deployment we think this protocol is actually for.

TABLE IV

THE KEY-COVERAGE GRADIENT. GROUPS DIFFER ONLY IN HOW MUCH OF THE QUESTION THE TRAINING SET TOUCHED. ρ = ACTIONS RELATIVE TO THE UNTRAINED FLAT STORE ON THE SAME QUESTIONS, SO LOWER IS CHEAPER; * MARKS A BOOTSTRAP 95% CI ON ρ EXCLUDING 1. $n = 30$ PER GROUP, IDENTICAL READER, ACTION SET AND BUDGET M=15. B3 INDEXES THE WHOLE CORPUS, FOR WHICH ALL FOUR GROUPS ARE THE SAME STORE, AND IS COMPARED SEPARATELY IN TABLE III.

Store	trained on	both keys	one key	neither
ρ: actions relative to B1				
B1 flat	1.000	1.000	1.000	1.000
B2 GraphRAG	0.909	0.984	1.036	0.947
Ours (trained)	**0.686***	**0.935**	**1.032**	**0.904***
F1				
B1 flat	0.700	0.600	0.767	**0.833**
B2 GraphRAG	0.633	0.567	0.833	0.800
Ours (trained)	**0.800**	**0.767**	**0.867**	**0.833**

TABLE V

INDEX CONSTRUCTION QUALITY, BY THE KIND OF KEY THE INDEX IS BUILT ON, WITH ONE EXAMPLE OF EACH.

Key the index is built on	n	deg.	prec.	recall
attribute: city *"Residents of Dorringham"*	8	26.4	98%	95%
attribute: hobby / job *"People whose hobby is astronomy"*	54	12.7	98%	96%
single-entity hub *"Bennett Coldwater"*	91	8.0	95%	90%
relation, two-hop *"Grandchildren of Ivor Yarrow"*	10	3.9	87%	55%
relation, one-hop *"Friends of Delphine Grimsby"*	54	3.4	78%	73%
all scored indexes	220	8.5	94%	91%

C. E3: Retention by Question Class

On the same 100 questions seen a second time, forward F1 rises $0.63 \rightarrow 0.82$ and the movement is entirely in the relation-level classes: the forward chains were saturated from the start (QC1–QC3 and QC9 at 1.00 in both epochs, since document-level retrieval already solves them), while counts QC4 go $0.33 \rightarrow 1.00$, joins QC6 $0.33 \rightarrow 0.92$, reverse lookup QC10 $0.20 \rightarrow 0.60$, intersection QC7 $0.67 \rightarrow 0.83$, deep counts QC8 $0.42 \rightarrow 0.58$ and abstention QC5 $0.00 \rightarrow 0.25$ (n between 5 and 14 per class). The gains land exactly in the classes retrieval cannot solve natively. Cost falls too — 7.4 steps and 70k tokens against 8.4 and 118k, budget exhaustion $15/100$ to $8/100$, and on the 83 questions resolved within budget in *both* epochs the lookup shortens $7.16 \rightarrow 6.37$ steps while F1 rises $0.759 \rightarrow 0.892$, so the saving is not an artifact of more questions terminating early. These two epochs are not a controlled store contrast, though: the store evolves *during* epoch 1, so the epoch label mixes store state with question order. The controlled version is the *trained* column of Table IV, which gives a larger saving ($\rho = 0.686$) precisely because it removes that mixing.

D. E4: What the Agent Built

Shape and cost of curation. Fig. 3 maps the store's evolution across its semantic space: structure grows from nothing into a navigation layer, hub indexes fanning out into topic clusters. Trace replay puts numbers on it (Fig. 4a): +287 documents (+4.9%) and 1,913 links, nearly all laid down in epoch 1. The backward pass is dominated by two actions — 307 `add` and 284 `link_many` calls across the run, against 3 edits, 12 deletes and 9 unlinks — so the agent builds and almost never revises. The deduplication judge fired on 45 candidates and merged 8; document-level parsimony was already holding at this scale.

Are they indexes? Calling every authored document an index would beg the question, so we classify the 287 by three checkable properties: at least one link; a sentence that does not enumerate its own members (the shape the curation skill forbids, tested by resolving the key's true member set from the universe and looking for those names in the text); and a resolvable key. On that test 242 (84.3%) are genuine indexes, 43 (15.0%) are empty stubs like *"Delphine Thistlewood"*, and 2 are materialized answers. That 84% is itself a result: in an earlier version of this environment the same protocol produced the opposite shape, 80% of authored documents stating the verified answer as a sentence, which serves the one question that produced it. The difference is that the curation skill now names the mechanism ("an index points; it does not list") rather than only the goal.

Accurate, but narrow. Table V scores every index whose key resolves. A link is correct if it points at a document about a genuine member of the key, resolved against the universe rather than by string match — the grandchildren of a person are recorded as "*X* is a child of *Y*" and never name the grandparent, so a string-match test scores a correct two-hop index at zero, and an earlier version of this analysis reported exactly that artifact. Resolved properly, precision is 94% and member recall 91% over the 220 whose key resolves. Quality tracks the arity of the key, not its depth: attribute indexes over a city or hobby are near-perfect (98%, 95–96% recall) and carry the most links, while relational indexes are smaller and noisier and two-hop relations are *not* worse than one-hop (87% vs. 78%). Depth is not what the curator struggles with; breadth is. Structurally the layer is flat: 1,896 of 1,913 links attach an index, 100 join two indexes and 17 join two source documents, so entry points are built readily and levels almost never — and 1,621 **of** 5,864 **source documents (**27.6%**) are reachable from an index in one read**. That is the binding constraint of Sec. VII-B, restated as a property of the store.

Break-even, and what would settle the mechanism. On the trained group the store saves 5,466 answering tokens per question (13.7k against 8.2k), so the 18.8M-token run repays itself after $N^* \approx 3{,}400$ questions (Eq. 2) — a figure that prices re-asking within the covered population, not generalization. We had intended to attribute the step savings to hits on built structure by splitting exam questions on whether the trajectory touched an index, and report that this split does not identify: touching is *downstream* of searching, so a longer trajectory is more likely to encounter an index and the touched group is

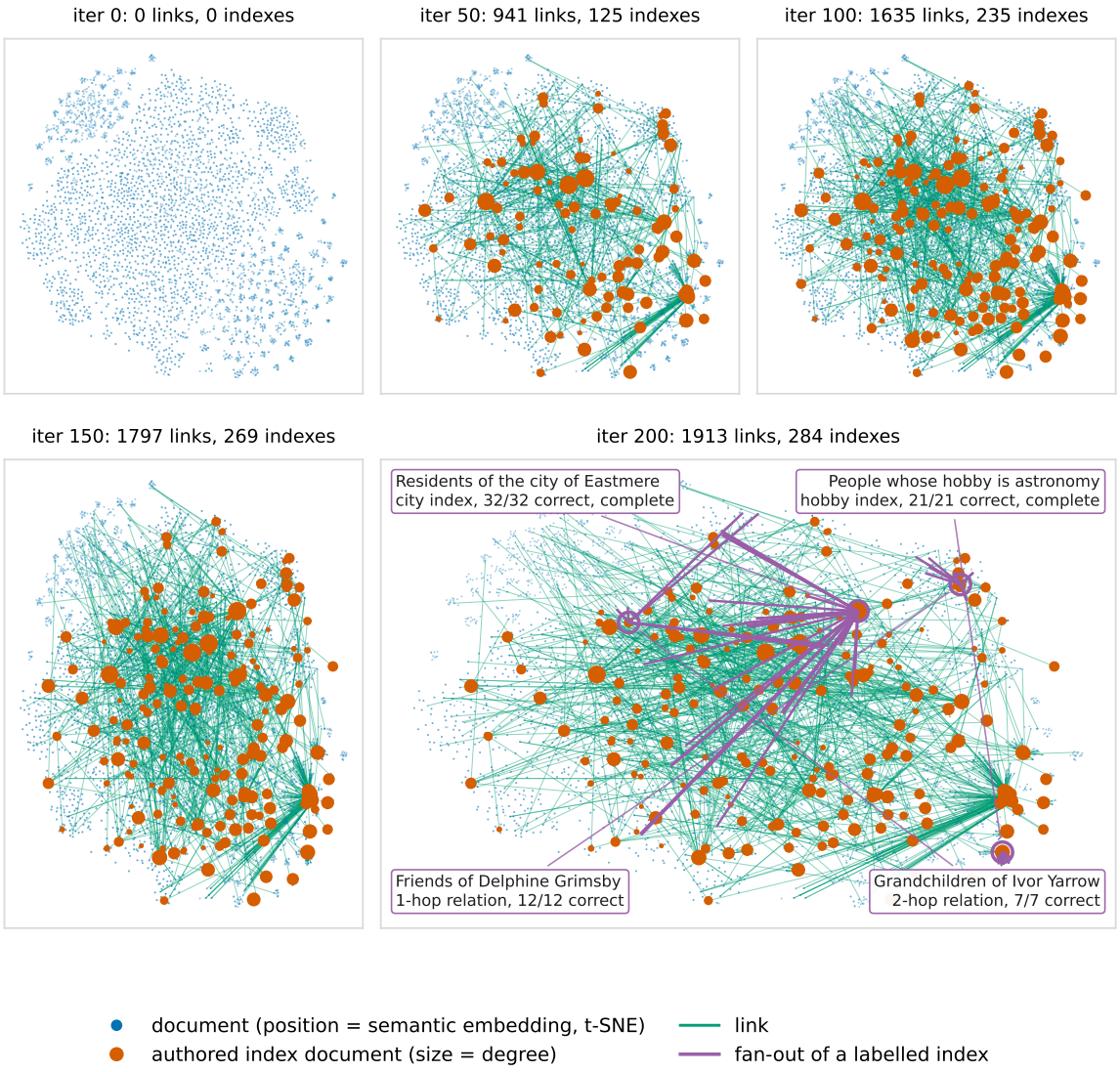

Fig. 3. The document network the agent built. *Top:* the store at three points in training; every document is drawn, and position is a t-SNE projection of its embedding under the same function the store indexes with, so documents drawn near each other are documents the reader's `search` retrieves together (t-SNE preserves neighbourhoods, not distances, so only local proximity should be read). One layout, computed once on the final store and keyed by document, is shared by all panels, so a document holds its position throughout and only the structure drawn over it changes; the clusters visible at iteration 0 are the universe's natural topic groups. Blue: source documents. Orange: authored index documents, sized by degree. Green: links. The final panel (bottom right, at two thirds width) carries four index documents labelled, together with the documents each one reaches (purple). The four were selected for correctness, not size: every link each of them carries points at a document that genuinely belongs under its key (Sec. VII-D), and together they span what the agent produced — an attribute index over a city, an attribute index over a hobby, a one-hop relation, and a two-hop relation. Structure grows from nothing into a navigation layer over the semantic space: hub indexes fan out into topic clusters while local links join semantically adjacent documents.

selected for difficulty by construction. The gradient probe is the intervention that does identify, because it varies coverage of the question's key *before* the reader starts.

VIII. Discussion and Limitations

Coverage, not construction, is the limit. The indexes the agent builds are accurate (94% precision, 91% member recall); what it does not build is *enough* of them, and the quarter of the corpus it covers is the quarter the training questions named. Two mechanisms are implicated, both actionable: the backward budget cannot populate a thirty-member index in one iteration and nothing asks the agent to *return* to one, so 43 were opened and abandoned; and more fundamentally the agent indexes the key a question names rather than the class it belongs to — "Friends of Delphine Grimsby" when asked about her, never "friends, for everyone". A protocol scoring an index by completeness over a class, and rewarding extension over creation, is the obvious next experiment. Curation has a boundary in the other direction too: GraphRAG-style summaries added to an already-searchable store *lose* accuracy, so the value is in knowing when not to edit — something

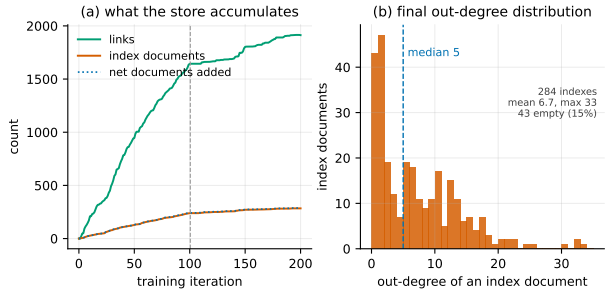

Fig. 4. (a) What the store accumulates, reconstructed by trace replay. *Index documents* = agent-authored documents still alive; *net documents added* = store size minus the initial 5,864. The two curves coincide: the agent adds access structure and essentially never rewrites source documents, which is the parsimony objective realized (growth $+4.9\%$ over 200 iterations). Link growth flattens after the epoch boundary (dashed), the store saturating on the keys the training questions name. (b) Out-degree of every index document in the final store. The distribution is long-tailed — median 5, mean 6.7, max 33 — with the right tail carrying the attribute indexes of Table V and a spike at zero: 43 indexes (15%) were opened and never filled.

downstream-only evaluation cannot see, since a store can be slowly ruined while individual answers still look fine.

Deployment and the online variant. The abstraction targets agent fleets maintaining shared repositories, where our failure ledger maps onto real incidents: near-duplicates burying each other is the lost-update problem, authored content without provenance is the hallucinated fix, a deleted last instance is knowledge loss during refactoring. We separate training from a frozen exam because that is what makes the generalization question answerable, but nothing in the method requires it: the online variant of Sec. VII-B needs a different *measurement*, not a different curator, since with the store moving underneath held-out accuracy stops being well defined and the honest alternative is prequential.

Threats to validity. Template-rendered language is simpler than natural prose and may flatter lexical matching; the PhantomWiki arm, whose generator we do not control, is the partial answer. Reader and trainer share a model family (`gpt-5-mini-2025-08-07`), so structure tuned by one may suit the other's habits — though a stronger reader is the harder test for us, not the easier one, since the better it is at recovering a set by searching the less an index adds. The four-point gradient is KBGym-only: it needs a generator that can enumerate a template's instance pool. Single seed; single agent by design; no support-set diagnostics on the external arm; grading is token-F1 against short golds.

References

[1] Retrieval as reasoning: self-evolving agent-native retrieval via LLM-wiki, arXiv:2605.25480, 2026.
[2] Streaming knowledge compilation: proactive materiality-scored pinning for time-evolving LLM wikis, arXiv:2606.09877, 2026.
[3] A. Karpathy, "llm-wiki," public note, Apr. 2026.
[4] D. Edge *et al.*, "From local to global: a GraphRAG approach to query-focused summarization," arXiv:2404.16130, 2024.
[5] P. Sarthi *et al.*, "RAPTOR: recursive abstractive processing for tree-organized retrieval," ICLR, 2024.
[6] B. Gutiérrez *et al.*, "From RAG to memory: non-parametric continual learning for LLMs," ICML, 2025.
[7] "Agentic context engineering: evolving contexts for self-improving language models," ICLR, 2026.
[8] A. Gong *et al.*, "PhantomWiki: on-demand datasets for reasoning and retrieval evaluation," ICML, 2025.
[9] H. Trivedi *et al.*, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," ACL, 2023.
[10] A. Asai *et al.*, "Self-RAG: learning to retrieve, generate, and critique through self-reflection," ICLR, 2024.
[11] "Reasoning RAG via system 1 or system 2: a survey on reasoning agentic retrieval-augmented generation," arXiv:2506.10408, 2025.
[12] W. Xu *et al.*, "A-MEM: agentic memory for LLM agents," arXiv:2502.12110, 2025.
[13] D. Wu *et al.*, "LongMemEval: benchmarking chat assistants on long-term interactive memory," ICLR, 2025; and LongMemEval-V2, arXiv:2605.12493, 2026.
[14] "MemDelta: controlled baselines and hidden confounds in agent memory evaluation," arXiv:2606.29914, 2026.
[15] Letta, "Sleep-time compute," 2025.
[16] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive NLP tasks," NeurIPS, 2020.
[17] V. Karpukhin *et al.*, "Dense passage retrieval for open-domain question answering," EMNLP, 2020.
[18] K. Guu *et al.*, "REALM: Retrieval-augmented language model pre-training," ICML, 2020.
[19] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," EACL, 2021.
[20] S. Yao *et al.*, "ReAct: Synergizing reasoning and acting in language models," ICLR, 2023.
[21] B. Jin *et al.*, "Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning," arXiv:2503.09516, 2025.
[22] C. Packer *et al.*, "MemGPT: Towards LLMs as operating systems," arXiv:2310.08560, 2023.
[23] P. Chhikara *et al.*, "Mem0: Building production-ready AI agents with scalable long-term memory," arXiv:2504.19413, 2025.
[24] J. S. Park *et al.*, "Generative agents: Interactive simulacra of human behavior," UIST, 2023.
[25] N. Shinn *et al.*, "Reflexion: Language agents with verbal reinforcement learning," NeurIPS, 2023.
[26] Z. Z. Wang *et al.*, "Agent workflow memory," arXiv:2409.07429, 2024.
[27] Y. Shao *et al.*, "Assisting in writing Wikipedia-like articles from scratch with large language models," NAACL, 2024.
[28] Z. Guo *et al.*, "LightRAG: Simple and fast retrieval-augmented generation," arXiv:2410.05779, 2024.
[29] B. Gutiérrez *et al.*, "HippoRAG: Neurobiologically inspired long-term memory for large language models," NeurIPS, 2024.
[30] Z. Yang *et al.*, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," EMNLP, 2018.
[31] H. Trivedi *et al.*, "MuSiQue: Multihop questions via single-hop question composition," TACL, 2022.
[32] X. Ho *et al.*, "Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps," COLING, 2020.
[33] P. Rajpurkar *et al.*, "SQuAD: 100,000+ questions for machine comprehension of text," EMNLP, 2016.
[34] A. Maharana *et al.*, "Evaluating very long-term conversational memory of LLM agents," ACL, 2024.
[35] K. Meng *et al.*, "Locating and editing factual associations in GPT," NeurIPS, 2022.
[36] K. Meng *et al.*, "Mass-editing memory in a transformer," ICLR, 2023.
[37] E. Zelikman *et al.*, "STaR: Bootstrapping reasoning with reasoning," NeurIPS, 2022.