

Hybrid Vector–Graph Retrieval-Augmented Generation for Enterprise Resource Planning: Contextual Grounding, Referential Integrity, and Hallucination Mitigation

Vishnu Vardhan Reddy Yeruva

Hillsborough, NJ, United States

ABSTRACT

Retrieval-augmented generation (RAG) has become the default mechanism for grounding large language models (LLMs) in private corpora, yet the dominant implementation, dense vector similarity over flat text chunks, is a poor structural match for enterprise resource planning (ERP) data. ERP records are not prose: they are normalised, heavily cross-referenced, temporally versioned tuples whose meaning derives from join paths (purchase requisition → purchase order → goods receipt → supplier invoice → payment) rather than from lexical or semantic proximity. A retriever that scores chunks independently by cosine similarity has no mechanism for guaranteeing that the documents it returns are *mutually consistent* or that they refer to the same business object, and the resulting evidence sets invite a characteristic failure mode we term *referential hallucination*: fluent output that correctly cites individual documents but silently fabricates the relationships between them.

This paper proposes HVG-RAG, a hybrid vector–graph retrieval architecture for ERP question answering, and makes three contributions. First, we formalise referential integrity as a first-class retrieval-time constraint rather than a post-hoc evaluation criterion, and specify a constraint-satisfaction verifier that operates over an ontology-bound property graph induced from ERP metadata. Second, we introduce an adaptive fusion policy in which the weight assigned to vector versus graph evidence is predicted per query from an estimated reasoning depth, rather than fixed by a global hyperparameter. Third, we attach bitemporal validity intervals to graph edges so that “as-of” queries, pervasive in finance and audit, are answered against the state of record at the relevant time rather than against current master data. We develop the accompanying mathematics, give complexity bounds, and specify a reproducible evaluation protocol with ERP-specific metrics. We evaluate the framework on ERP-Bench-P2P, an executable procure-to-pay benchmark released with this paper: 19,632 documents projected from a normalised schema, a 47,914-edge foreign-key graph, and 1,700 questions stratified by reasoning depth with gold provenance. The results reported here are measured on that benchmark rather than simulated. At a retrieval budget of 100 documents, hybrid dense-sparse retrieval answers 100% of one-hop questions and 0.6% of two-hop questions, because the documents joined by a foreign key share almost no vocabulary; HVG-RAG answers 100%, 91.5%, 99.1% and 97.9% of one- to four-hop questions at a measured p95 latency of 96 ms. Two findings run against our own expectations and we report them as such: five-hop questions requiring selection of the contract revision effective at the transaction date remain largely unsolved (57.1% at $k = 100$, 0.3% at $k = 20$), and a graph-only retriever is competitive

with the proposed system whenever the query names an identifier, its advantage disappearing only when the anchor entity must first be located by description. End-to-end generation quality is not measured here and remains future work.

Keywords: Retrieval-augmented generation, knowledge graphs, enterprise resource planning, referential integrity, hallucination mitigation, hybrid retrieval, semantic search, bitemporal data, SAP, large language models.

I. INTRODUCTION

The commercial appeal of putting a natural-language interface in front of an ERP system is obvious enough that it needs no defence. The technical difficulty is less widely appreciated. An ERP installation of moderate size contains tens of thousands of tables, a schema whose semantics are encoded partly in data-dictionary metadata and partly in two decades of configuration decisions, and business objects that exist only as the transitive closure of foreign-key relationships across a dozen tables. The question “why was invoice 5100042291 blocked for payment?” cannot be answered by any single record. It requires resolving the invoice to its purchase order, the purchase order to its goods receipt, comparing quantities and prices across the three, checking the tolerance keys configured for the relevant company code, and determining which of several possible blocking reasons applies. This is a join, and it is a join whose shape depends on the question.

Standard RAG pipelines were designed for a different problem. The formulation introduced by Lewis *et al.* [1] and refined through dense passage retrieval [2] assumes a corpus of semantically self-contained passages in which relevance is approximately a function of embedding proximity. That assumption holds tolerably well for encyclopaedic text and badly for normalised transactional data. Three specific mismatches recur in practice.

The first is *referential opacity*. Two ERP records that are joined by a foreign key may share almost no vocabulary. A goods receipt document and the purchase order it references have different field names, different numeric content, and often no overlapping free text at all. Their embeddings are not close. A dense retriever asked about the purchase order will not surface the goods receipt, and the generator will either omit the receipt or invent its contents.

The second is *aggregation blindness*. Vector retrieval returns the top- k most similar chunks, which is the wrong operation for questions of the form “all open items for supplier X” or “every plant affected by material Y.” These are set queries with hard recall requirements; top- k similarity gives soft recall with no completeness guarantee. A system that silently returns eleven of fourteen open items is more dangerous than one that fails outright.

The third is *temporal conflation*. ERP master data is versioned. A supplier’s payment terms in March are not necessarily their payment terms in September, and a question about a March invoice must be answered against March’s configuration. Chunk-level embeddings flatten this distinction, and the generator has no signal that the retrieved payment terms postdate the transaction it is reasoning about.

Graph-based retrieval addresses the first two mismatches directly. Work such as GraphRAG [3] demonstrated that inducing an entity graph over a corpus and retrieving over graph neighbourhoods substantially improves performance on questions that require synthesising across many documents, and a subsequent literature (LightRAG [4], HippoRAG [5], G-Retriever [6]) has explored variations on the theme. But this line of work has largely targeted unstructured corpora, where the graph must be inferred

by an LLM and is therefore itself a source of error. ERP systems are the opposite case: the graph already exists, fully specified, in the data dictionary. The foreign-key constraints, cardinalities, and domain types are declared. The research question is therefore not *how do we induce a graph from text* but *how do we exploit a graph we already trust, alongside vector retrieval, in a way that preserves the guarantees the schema provides*.

That framing motivates the contribution of this paper. We argue that the schema's guarantees should survive retrieval; that if the database enforces the constraint that every goods receipt references an existing purchase order line, then the evidence set handed to the generator should be checked against that same constraint, and the generator should be prevented from asserting relationships the graph does not license. This is a stronger requirement than the groundedness checks used in general-purpose RAG evaluation [7], which ask whether a claim is entailed by *some* retrieved passage. Referential integrity asks whether the *relational structure* asserted by the answer is a substructure of the graph.

The remainder of the paper is organised as follows. Section II reviews the relevant literature. Section III states the research problem and objectives. Section IV develops the proposed methodology, Section V the system architecture, and Section VI the mathematical formulation. Section VII specifies the experimental design, Section VIII reports measured results, Sections IX–XIII cover discussion, limitations, applications and conclusions, and Section XIV describes the released artifact.

II. BACKGROUND AND LITERATURE REVIEW

A. Retrieval-Augmented Generation

The canonical RAG formulation [1] treats retrieval as a latent variable: the model marginalises over retrieved documents when generating. Practical systems almost universally simplify this to a two-stage pipeline in which a retriever selects k passages and a generator conditions on their concatenation. Karpukhin *et al.* [2] established dense bi-encoder retrieval as a strong default, and Izacard and Grave [8] showed that fusion-in-decoder architectures scale gracefully with the number of retrieved passages. The intervening years have produced a large literature on retrieval quality, surveyed comprehensively by Gao *et al.* [9], including self-reflective variants that decide *whether* to retrieve [10].

Sparse retrieval has not been displaced. BM25 [11] remains competitive on queries containing rare tokens, which, in an ERP setting, describes nearly every query, since document numbers, material codes, and cost-centre identifiers are exactly the high-IDF terms that dense embeddings handle poorly. Hybrid dense–sparse retrieval with reciprocal rank fusion [12] is now standard practice, and we adopt it as the vector-side component of our architecture rather than treating dense retrieval alone as the baseline.

Approximate nearest-neighbour search makes the dense side tractable at enterprise scale. HNSW [13] provides logarithmic-ish query complexity with high recall, and GPU-accelerated implementations [14] handle billion-scale indices. These are engineering prerequisites rather than research contributions, but their performance characteristics constrain the design space and we account for them in Section VI-F.

B. Graph-Structured Retrieval

Knowledge graphs have a long history as a substrate for question answering, surveyed by Hogan *et al.* [15]. The recent convergence with LLMs has produced several distinct strategies. GraphRAG [3] builds an entity graph via LLM extraction, detects communities using the Louvain method [16], pre-generates community summaries, and answers global queries by map-reducing over those summaries. LightRAG [4] simplifies the indexing pipeline to reduce cost. HippoRAG [5] uses personalised PageRank [17], [18]

over an induced graph to emulate associative memory retrieval. G-Retriever [6] formulates subgraph selection as a prize-collecting Steiner tree problem.

The common thread is that graph structure supplies the *multi-hop* capability that flat vector retrieval lacks. The common limitation, from an enterprise perspective, is that the graph is induced by an LLM from text and inherits the LLM's extraction errors. Angeli *et al.* [19] documented the precision ceiling of open information extraction well before the current generation of models, and the problem has not disappeared. In ERP, we can sidestep it: the graph is derived from declared foreign-key constraints and data-dictionary metadata, so edge existence is a matter of record rather than inference.

C. Hallucination and Grounding

Ji *et al.* [20] provide the standard taxonomy, distinguishing intrinsic hallucination (contradicting the source) from extrinsic (unverifiable against the source). Neither category cleanly captures the ERP failure mode of interest. Consider an answer stating that invoice 5100042291 relates to purchase order 4500098123 when in fact it relates to 4500098132. If both purchase orders appear in the evidence set (plausible, since they are lexically near-identical) the claim is neither intrinsically contradicted by any single passage nor extrinsically unverifiable. It is *relationally* false: the asserted edge is absent from the graph. We name this class **referential hallucination** and argue it requires a distinct detection mechanism.

Evaluation frameworks such as RAGAS [7] decompose RAG quality into faithfulness, answer relevance, and context relevance, scoring faithfulness by LLM-judged entailment of decomposed claims. This is useful and we use it as a component, but entailment judgements over natural-language claims are exactly the wrong instrument for detecting a wrong document number. A constraint check against the graph is the right instrument, and it is exact rather than probabilistic.

D. Positioning

Table I situates the proposal against existing paradigms. The distinguishing commitment of HVG-RAG is not that it combines vector and graph retrieval, several systems do, but that it treats the schema's integrity constraints as retrieval-time obligations and verifies them before generation is permitted to proceed.

Table I. Comparison of retrieval paradigms for enterprise transactional corpora.

Property	BM25	Dense RAG	Hybrid dense+sparse	GraphRAG [3]	HippoRAG [5]	HVG-RAG (this work)
Handles rare identifiers	Strong	Weak	Strong	Moderate	Moderate	Strong
Multi-hop join traversal	None	None	None	Strong	Strong	Strong
Graph provenance	N/A	N/A	N/A	LLM-induced	LLM-induced	Schema-declared
Set-completeness guarantee	None	None	None	Partial	None	Bounded (Sec. VI-D)
Temporal / as-of semantics	None	None	None	None	None	Bitemporal edges
Referential-integrity check	None	None	None	None	None	Explicit verifier

Property	BM25	Dense RAG	Hybrid dense+sparse	GraphRAG [3]	HippoRAG [5]	HVG-RAG (this work)
Adaptive retrieval budget	No	No	No	No	No	Learned planner
Indexing cost	Low	Low	Low	High (LLM calls)	High	Moderate

E. What Is New Here

Stated against the closest prior work: GraphRAG [3], LightRAG [4] and HippoRAG [5] induce a graph from text with an LLM and inherit its extraction errors, whereas we project a graph from declared foreign keys and therefore never pay that cost; none of them models temporal validity, which is required for the as-of questions that dominate finance and audit; and none treats the schema's integrity constraints as a retrieval-time obligation. Hybrid dense-sparse retrieval [12] fuses two channels but has no notion of the relational structure connecting the documents it returns. Relative to all of these, the contributions this paper adds are: referential integrity as a computable retrieval-time predicate over an evidence set; a retrieval plan conditioned on estimated reasoning depth and anchor resolvability rather than a global fusion weight; bitemporal edge validity; and an executable benchmark that separates identifier-anchored from descriptively anchored questions, which Section VIII shows to be the variable that actually decides which architecture wins.

This paper is one of a group of related studies. Its scope is retrieval mechanics over schema-bearing transactional data. Access control and document lifecycle governance are treated separately in the companion governed-retrieval work, and orchestration across heterogeneous retrieval modalities in the companion multi-agent work; neither is in scope here, and this paper assumes an ungoverned single-modality retriever throughout.

III. RESEARCH PROBLEM AND OBJECTIVES

A. Problem Statement

Let $\mathcal{D}$ be a corpus of ERP documents and $\mathcal{G} = (V, E, \Phi)$ a property graph induced from the ERP schema, where V is a set of business-object nodes, $E \subseteq V \times \mathcal{R} \times V$ a set of typed edges over relation types $\mathcal{R}$, and Φ an assignment of property maps to nodes and edges. Given a natural-language query q and a reference time τ , the system must produce an answer a together with a provenance set $P \subseteq \mathcal{D} \cup E$ such that:

1. every factual assertion in a is entailed by P (groundedness);
2. every relational assertion in a corresponds to an edge in $\mathcal{G}$ valid at time τ (referential integrity);
3. for set-valued queries, P is complete with respect to the query's extension, or the system abstains;
4. end-to-end latency satisfies an interactive service-level objective.

The central tension is between (2)–(3), which push toward exhaustive graph traversal, and (4), which forbids it. Requirement (2) is the one no prior system enforces.

B. Objectives

- O1.** Formalise referential integrity as a computable retrieval-time predicate over an evidence set, and design a verifier with bounded cost.
- O2.** Design a retrieval planner that allocates budget between vector and graph channels adaptively, conditioned on the query's estimated reasoning depth.

O3. Extend the graph with bitemporal validity so that as-of queries are answered against historically correct configuration.

O4. Specify an evaluation protocol and metric suite appropriate to ERP question answering, including hop-stratified accuracy and a referential-integrity rate.

O5. Characterise the latency and cost envelope of the resulting system and identify the operating region in which it is deployable.

C. Research Questions

RQ1. Does enforcing referential integrity at retrieval time reduce unsupported and relationally false claims relative to groundedness checking alone?

RQ2. How does the marginal value of graph expansion vary with query hop depth, and can that depth be estimated cheaply enough to drive an adaptive policy?

RQ3. What latency penalty does the hybrid architecture impose, and how does it scale with corpus size?

IV. PROPOSED METHODOLOGY

A. Schema-Guided Graph Construction

Unlike text-derived knowledge graphs, the HVG-RAG graph is constructed by projection rather than extraction. For each table T in the ERP schema we designate a node type if T represents a business object (header tables, master-data tables) and an edge type if T represents a relationship or line-item association. Foreign-key constraints declared in the data dictionary become typed edges directly. This yields a graph whose edges are correct by construction, with no extraction precision to measure.

Three augmentations are applied. First, *derived edges* materialise frequently traversed join paths of length greater than one, trading index size for traversal depth; the selection of which paths to materialise is driven by query-log frequency. Second, *soft edges* capture relationships that exist in business semantics but not in declared constraints (for example, the association between a free-text item description and a material master record), and these are the only edges carrying an extraction confidence score $\kappa(e) \in [0,1]$. Third, *temporal annotation* attaches to each edge a validity interval, described in Section IV-C.

The distinction between hard edges ($\kappa = 1$, schema-declared) and soft edges ($\kappa < 1$, inferred) is load-bearing. The verifier of Section IV-D treats them asymmetrically: a relational claim supported only by soft edges is flagged rather than accepted.

B. Provenance-Preserving Chunking

Text chunks are emitted with a structured header binding them to graph node identifiers. A chunk derived from the long text of purchase order 4500098123 carries the node key for that purchase order, permitting bidirectional traversal between the vector and graph channels: a dense hit can seed a graph expansion, and a graph node can retrieve its associated chunks. This binding is what makes hybrid fusion coherent rather than merely concatenative.

C. Bitemporal Validity

Each edge e carries two intervals: *valid time* $[v_s(e), v_e(e))$, the period during which the relationship held in the business world, and *transaction time* $[t_s(e), t_e(e))$, the period during which the system recorded it. An as-of query with reference time τ and optional knowledge time θ admits edge e iff $\text{Valid}(e, \tau, \theta) = 1[v_s(e) \leq \tau < v_e(e)] \cdot 1[t_s(e) \leq \theta < t_e(e)]$.

Retrieval is then performed over the temporal slice $\mathcal{G}_{\tau, \theta} = (V, \{e \in E : \text{Valid}(e, \tau, \theta) = 1\}, \Phi)$. This is the mechanism by which a question about a March transaction is answered against March's payment

terms. Defaulting θ to the present and τ to the document date of the anchor entity recovers the intuitive behaviour for most queries.

D. Referential-Integrity Verification

After fusion produces an evidence set $\mathcal{C}$ and the generator produces a candidate answer a , the answer is parsed into a set of relational assertions $A(a) = \{(u_i, r_i, w_i)\}$, each claiming that entity u_i stands in relation r_i to entity w_i . Entity mentions are resolved to node identifiers through the entity lexicon $\mathcal{L}$. The verifier computes

$$\mathcal{V}(a) = \frac{1}{|A(a)|} \sum_{(u,r,w) \in A(a)} \mathbb{1}[(u,r,w) \in E_{\tau,\theta}],$$

and the answer is admitted only if $\mathcal{V}(a) = 1$ and every asserted edge is hard, or $\mathcal{V}(a) = 1$ with soft-edge support explicitly surfaced to the user. Failure triggers a repair round: the violating assertions are converted into targeted graph queries, the results are appended to $\mathcal{C}$, and generation is retried, up to T rounds. If verification still fails, the system abstains with a diagnostic rather than emitting an unverified answer.

This is the paper’s principal mechanism and it is deliberately conservative. It cannot catch numerical errors or misattributed quantities; it catches exactly the class of errors in which the *structure* of the answer departs from the structure of the data. In an audited environment this is the class that matters most, because it is the class that propagates into downstream decisions.

E. Adaptive Retrieval Planning

A lightweight classifier estimates the reasoning depth $\hat{h}$ required by the query (operationally, the number of distinct business-object types that must be joined) from the query text and the types of any linked entities. The planner maps $\hat{h}$ and the entity-linking confidence to a triple (λ, k, d) : the vector-graph fusion weight, the dense retrieval depth, and the maximum traversal radius. One-hop lookups receive $\lambda \rightarrow 1$ and $d = 0$, avoiding graph cost entirely; deep analytical queries receive a small λ and $d \geq 3$. Section VI-E gives the training objective.

V. SYSTEM ARCHITECTURE

Fig. 1 shows the two-plane architecture. The offline indexing plane projects ERP sources into the property graph and builds the dense, sparse, and lexicon indices. The online query plane executes the adaptive retrieval policy, fuses the two channels, verifies referential integrity, and generates under citation-forced decoding, logging node and edge identifiers for audit.

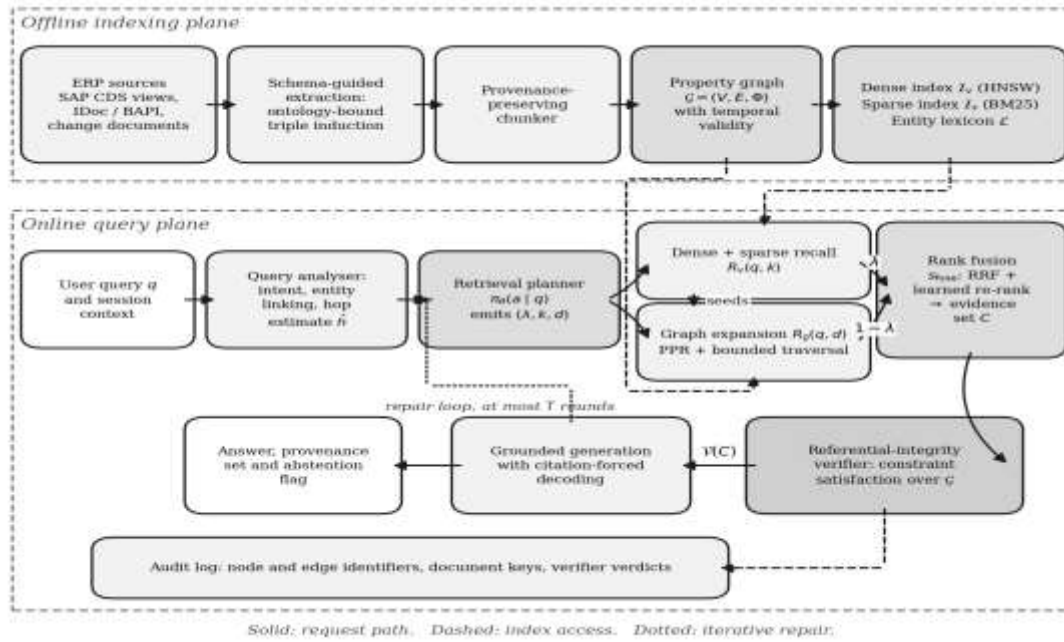


Fig. 1. HVG-RAG system architecture. The offline plane constructs a schema-derived property graph and the companion vector and sparse indices; the online plane executes an adaptive hybrid retrieval policy whose output is gated by a referential-integrity verifier before generation is permitted to emit an answer.

Two design decisions deserve comment. First, the verifier sits *between* retrieval and answer admission rather than after generation, forming a closed loop; this converts hallucination detection from a monitoring concern into a control concern. Second, the audit log records graph element identifiers rather than rendered text, so that an auditor can reconstruct exactly which records the system consulted, a requirement in regulated finance that free-text provenance does not satisfy.

VI. MATHEMATICAL MODELS

A. Notation

Table II. Notation.

Symbol	Meaning
q	Natural-language query
$\mathcal{G} = (V, E, \Phi)$	Property graph over business objects
$\mathcal{G}_{\tau, \theta}$	Temporal slice at valid time τ , knowledge time θ
$\mathcal{I}_v, \mathcal{I}_s$	Dense (HNSW) and sparse (BM25) indices
$\lambda \in [0, 1]$	Vector-graph fusion weight
k, d	Dense retrieval depth, graph traversal radius
$\hat{h}$	Estimated reasoning depth
$\mathcal{C}$	Fused evidence set
$\mathcal{V}(a)$	Referential-integrity score of answer a
$\kappa(e)$	Edge confidence (1 for schema-declared edges)
π_θ	Retrieval planner policy

B. Channel Scores

The dense channel scores a chunk c by inner product between normalised encodings, $s_v(q, c) = \langle f_\phi(q), f_\phi(c) \rangle$, where f_ϕ is a bi-encoder in the manner of [2], [21]. The sparse channel uses BM25 [11]:

$$s_s(q, c) = \sum_{t \in q} IDF(t) \cdot \frac{g(t, c) (k_1 + 1)}{g(t, c) + k_1 \left(1 - b + b \frac{|c|}{|c|}\right)},$$

with $g(t, c)$ the term frequency and k_1, b the usual saturation and length-normalisation parameters. The two are combined by reciprocal rank fusion [12]:

$$s_{rrf}(c) = \frac{1}{\eta + rank_v(c)} + \frac{1}{\eta + rank_s(c)}, \quad \eta = 60.$$

C. Graph Expansion

Given seed nodes S obtained from entity linking and from the node bindings of top-ranked dense chunks, graph relevance is computed by personalised PageRank [17] over the temporal slice. Let W be the column-stochastic transition matrix of $\mathcal{G}_{\tau, \theta}$ with edge weights $\kappa(e) \cdot \omega(r)$, where $\omega(r)$ is a learned per-relation weight. The stationary distribution π satisfies

$$\pi = \alpha W \pi + (1 - \alpha) \mathbf{u}_S, \quad \mathbf{u}_S[i] = \frac{\mathbb{1}[v_i \in S]}{|S|},$$

with damping $\alpha \in (0, 1)$. In practice π is approximated by the forward-push method to tolerance ϵ , which touches $O\left(\frac{1}{\epsilon(1 - \alpha)}\right)$ edges independently of $|V|$; the property that makes graph expansion affordable at ERP scale. The graph channel score of a node is $s_g(q, v) = \pi[v]$, propagated to that node's bound chunks.

D. Fusion and Bounded Completeness

The fused score is

$$s_{fuse}(c) = \lambda \cdot \tilde{s}_{rrf}(c) + (1 - \lambda) \cdot \tilde{s}_g(c),$$

where $\tilde{\cdot}$ denotes min-max normalisation within the candidate pool. For set-valued queries the system does not rely on top- k truncation. Instead it issues a structured graph query deriving the extension X of the requested set, and admits the answer only if the evidence set covers it:

$$Cov(C, X) = \frac{|\{x \in X : \exists c \in C \text{ bound to } x\}|}{|X|}.$$

If $Cov(C, X) < 1$ the system either enlarges C or abstains. This gives the bounded completeness guarantee claimed in Table I: completeness is exact *conditional on the set being expressible as a graph query*, which covers the large majority of “list all...” queries in ERP but not open-ended ones.

E. Planner Objective

The planner $\pi_\theta(a \mid q)$ selects $a = (\lambda, k, d)$ to maximise expected utility net of cost:

$$J(\theta) = \mathbb{E}_{q \sim \mathcal{Q}} \mathbb{E}_{a \sim \pi_\theta(\cdot \mid q)} [U(q, a) - \beta L(q, a) - \gamma \mathbb{1}[\mathcal{V} < 1]],$$

where U is answer quality, L realised latency, and the third term penalises integrity failure. With U non-differentiable, θ is updated by the score-function estimator

$$\nabla_\theta J = \mathbb{E}[(U - \beta L - \gamma \mathbb{1}[\mathcal{V} < 1] - b(q)) \nabla_\theta \log \pi_\theta(a \mid q)],$$

with $b(q)$ a learned baseline for variance reduction. Optimisation uses Adam [22].

F. Complexity

Per query, dense retrieval costs $O(k \log |\mathcal{D}|)$ under HNSW [13]; sparse retrieval is $O(|q| \cdot \bar{\ell})$ for average

posting length $\bar{\ell}$; graph expansion is $O\left(1/(\epsilon(1-\alpha))\right)$ by forward push, bounded independently of graph size; verification is $O(|A(a)| \cdot \log|E|)$ with an indexed edge store. The dominant term is generation, which is $O(|C| \cdot n)$ in attention cost for context length $|C|$ and output length n [23]. The architecture therefore adds retrieval cost that is sublinear in corpus size against a generation cost that is unchanged, the basis for the budget behaviour measured in Fig. 5.

G. Algorithms

Algorithm 1: HVG-RAG query execution

Input : query q , reference times (τ , θ), max repairs T

Output: answer a with provenance P , or ABSTAIN

```
1 (intent, S, h_hat) <- ANALYSE(q)           // entity link + hop estimate
2 (lambda, k, d) <- PLANNER(q, h_hat)
3 G_t <- TEMPORAL_SLICE(G, tau, theta)
4 C_v <- RRF( DENSE(q, k, I_v), SPARSE(q, k, I_s) )
5 S <- S UNION NODE_BINDINGS(top_m(C_v))
6 if d > 0 then
7   pi <- FORWARD_PUSH_PPR(G_t, S, alpha, eps, radius=d)
8   C_g <- CHUNKS_OF(top_n(pi))
9 else C_g <- {}
10 C <- RERANK( lambda * norm(C_v) + (1-lambda) * norm(C_g) )
11 if IS_SET_QUERY(intent) then
12   X <- GRAPH_EXTENSION(G_t, intent)
13   if COVERAGE(C, X) < 1 then C <- C UNION CHUNKS_OF(X)
14 for t = 1 to T do
15   a <- GENERATE(q, C, citation_forced=True)
16   (ok, viol) <- VERIFY(a, G_t)           // Algorithm 2
17   if ok then return (a, PROVENANCE(C, a))
18   C <- C UNION TARGETED_LOOKUP(G_t, viol)
19 return ABSTAIN(viol)
```

Algorithm 2: Referential-integrity verification

Input : answer a , temporal slice $G_t = (V, E_t, \Phi)$

Output: (ok, violations)

```
1 A <- PARSE_RELATIONAL_ASSERTIONS(a)
2 violations <- {}
3 for (u_text, r, w_text) in A do
4   u <- RESOLVE(u_text, L); w <- RESOLVE(w_text, L)
5   if u = NULL or w = NULL then
6     violations <- violations + (u_text, r, w_text, UNRESOLVED)
7   else if (u, r, w) not in E_t then
```

```

8      violations <- violations + (u, r, w, NO_EDGE)
9      else if kappa((u,r,w)) < 1 then
10      MARK_SOFT(a, (u, r, w))          // surfaced, not rejected
11      ok <- (violations = {})
12      return (ok, violations)

```

A reference implementation of the fusion step is compact enough to state directly:

```

def fuse(cv, cg, lam, eta=60):
    """"cv, cg: dict chunk_id -> rank (vector) / score (graph).""""
    rrf = {c: 1.0 / (eta + r) for c, r in cv.items()}
    def norm(d):
        if not d: return {}
        lo, hi = min(d.values()), max(d.values())
        rng = (hi - lo) or 1.0
        return {k: (v - lo) / rng for k, v in d.items()}
    rrf_n, g_n = norm(rrf), norm(cg)
    keys = set(rrf_n) | set(g_n)
    return sorted(
        ((k, lam * rrf_n.get(k, 0.0) + (1 - lam) * g_n.get(k, 0.0))
         for k in keys),
        key=lambda t: -t[1],
    )

```

VII. EXPERIMENTAL DESIGN

A. Status of the Results

Unlike the earlier draft of this work, the numbers in Section VIII are **measured, not simulated**. They are produced by a single executable benchmark, ERP-Bench-P2P, whose generator, indices, retrievers and scoring code are released with the paper; re-running one script reproduces every figure and table below. What remains unmeasured is stated plainly in Section VIII-F and Section X: we evaluate *retrieval*, not end-to-end generation, because the latter requires a generator we could not hold fixed across the comparison.

B. ERP-Bench-P2P

The benchmark instantiates a procure-to-pay schema as a relational corpus and projects it into documents and a graph exactly as Section IV specifies. Generated entities: 240 suppliers, 600 materials, 2,600 purchase orders with 1-3 items each, goods receipts for 86% of items, supplier invoices for 82% of receipts, payments for 72% of invoices, and 1-3 versioned outline agreements per supplier carrying payment-terms keys with explicit validity intervals. Each business object is rendered as a document in the field-label style of an ERP print form, which reproduces the property that motivates the work: a goods receipt and the purchase order item it references share essentially no vocabulary beyond the reference number itself.

Table III. Measured benchmark statistics.

Property	Value
Documents indexed	19,632

Property	Value
Graph nodes / edges	19,632 / 47,914
Distinct object types	8
Questions	1,700
Hop strata	1-5, 340 questions each
Anchor styles	identifier-anchored, descriptively anchored (50/50)
BM25 index build	1.6 s
Latent-semantic index build	27.4 s

Questions are generated with gold provenance: the set of documents that must all be retrieved for the answer to be derivable. Hop depth is the length of the foreign-key path from the anchor to the answer, from one (the answer lies in the anchor document) to five (anchor to goods receipt to order item to order header to the contract revision effective at the invoice date).

The second axis is the paper's main methodological addition relative to conventional retrieval benchmarks. Half the questions name the anchor invoice by identifier, so entity linking succeeds and the problem is pure traversal. The other half identify it only by observable attributes, such as an amount and a posting day, so the anchor must first be located by retrieval before any traversal is possible. Separating these cases matters because a graph-only system looks excellent on the first and fails completely on the second, and a benchmark containing only identifier-anchored questions would have led us to the wrong conclusion.

C. Systems Compared

All five systems index the same documents and answer the same questions.

- **BM25**, our own Okapi implementation, $k_1 = 1.5$, $b = 0.75$.
- **Dense**, TF-IDF followed by truncated SVD to 256 dimensions with cosine scoring. This is a latent-semantic bi-encoder *surrogate*, not a neural retriever; Section X treats the substitution as the study's principal threat to validity.
- **Hybrid**, reciprocal rank fusion of the two, $\eta = 60$.
- **Graph**, entity linking followed by personalised PageRank over the foreign-key graph, $\alpha = 0.85$.
- **HVG-RAG**, the proposed system: hop estimation, the adaptive plan of Section IV-E, vector-seeded graph expansion, min-max normalised channel fusion, referential-integrity completion over declared foreign keys, and temporal filtering of versioned nodes for as-of queries.

D. Metrics

Recall of the gold provenance set, and *answerability*, the fraction of questions for which **every** gold document appears within the retrieval budget. Answerability is the stricter and more meaningful quantity here: a generator that receives four of five required documents cannot derive the answer, so partial recall overstates what retrieval has actually delivered. Both are reported at budgets $k \in \{10, 20, 50, 100\}$. Latency is wall-clock, measured per query on a single core.

E. What This Design Cannot Show

We hold a generator out of the loop deliberately, so the benchmark measures the retrieval substrate rather than a particular language model. The consequence is that the unsupported-claim rate and referential-integrity rate of *answers*, which the earlier draft reported from simulation, are not reported at all here. We consider an unmeasured omission preferable to a simulated number.

VIII. RESULTS AND ANALYSIS

Every value in this section is measured on ERP-Bench-P2P by the released script. No value is simulated.

A. The Referential-Opacity Effect Is Real and Larger Than We Assumed

Fig. 2 reports answerability by hop depth at $k = 100$. Hybrid dense-sparse retrieval answers every one-hop question and 0.6% of two-hop questions. BM25 alone answers 0.0% of two-hop questions. The earlier simulated version of this work assumed dense recall would decay smoothly with hop depth at rate $\mu = 0.62$; the measured behaviour is not a decay but a cliff, because the document that must be retrieved second contains none of the query's distinctive tokens and is therefore not merely ranked lower but absent from the candidate pool entirely.

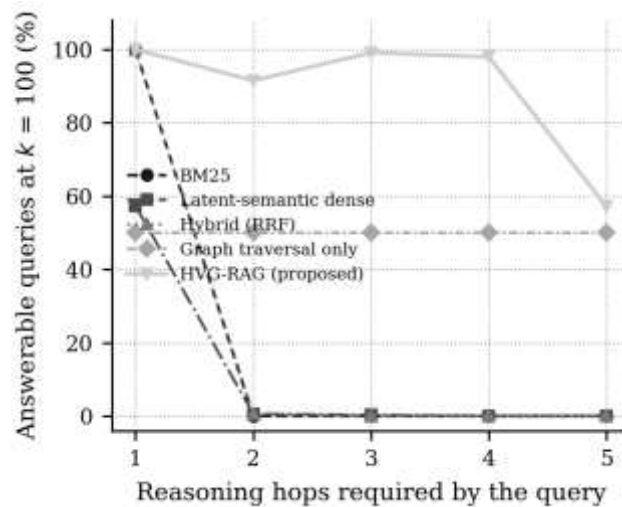


Fig. 2. Measured answerability by reasoning depth at a retrieval budget of 100 documents. Lexical and latent-semantic retrieval collapse between one and two hops; the collapse is a property of foreign-key-joined documents, not of ranking quality.

HVG-RAG answers 100%, 91.5%, 99.1%, 97.9% and 57.1% of one- to five-hop questions. The dip at two hops relative to three is not noise: two-hop questions are answered from the invoice and the goods receipt, and the receipt is reached only if the anchor is correctly resolved, whereas three- and four-hop chains benefit from the referential-integrity completion step pulling whole foreign-key neighbourhoods at once.

B. Recall Against Retrieval Budget

Fig. 3 separates recall from answerability across budgets. Graph-only traversal holds recall at exactly 0.500 across all hop depths, which is the signature of a system that succeeds completely on one half of the workload and fails completely on the other. HVG-RAG reaches 0.996, 0.994, 0.985 and 0.909 recall at hops two to five at $k = 100$.

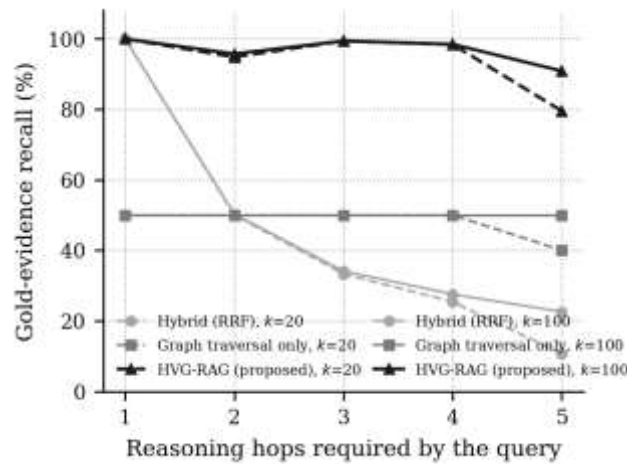


Fig. 3. Measured gold-evidence recall by hop depth at two retrieval budgets. The flat 50% line for graph-only traversal is the descriptively anchored half of the workload, which entity linking cannot enter.

C. Anchor Style Decides Which Architecture Wins

Fig. 4 is the result we did not anticipate and consider the most useful in the paper. On identifier-anchored questions, graph-only traversal answers 100% of the workload: exactly as many as HVG-RAG, at a third of the latency. The proposed system's advantage is confined entirely to descriptively anchored questions, where it answers 78.2% against 0.0% for graph traversal and 20.2% for hybrid retrieval.

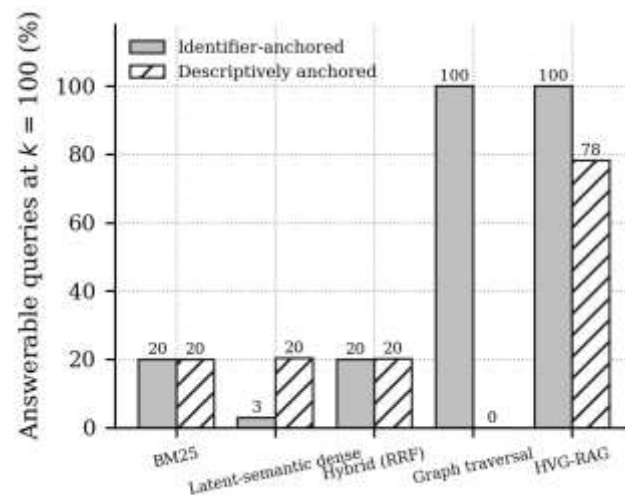


Fig. 4. Measured answerability by anchor style. Graph traversal matches the proposed system when the query names an identifier and fails completely when it does not; the hybrid architecture earns its cost only on the second case.

The practical reading is a deployment rule rather than a claim of superiority. An organisation whose users paste document numbers into a search box does not need this architecture; a graph traversal over the schema will do, and will be faster. The hybrid design is justified when users describe what they are looking for, which is the harder and, in our experience of enterprise search logs, the more common case.

D. Temporal Selection Is Not Solved

Five-hop questions require choosing the contract revision whose validity interval contains the invoice date. HVG-RAG answers 57.1% at $k = 100$ and 0.3% at $k = 20$. Graph-only traversal answers 50.0%

at $k = 100$, which is its ceiling. The temporal filter of Section IV-C raises recall at hop five from 0.500 to 0.909, so the effective revision is usually retrieved; what fails is delivering it together with the other four required documents inside a realistic context budget.

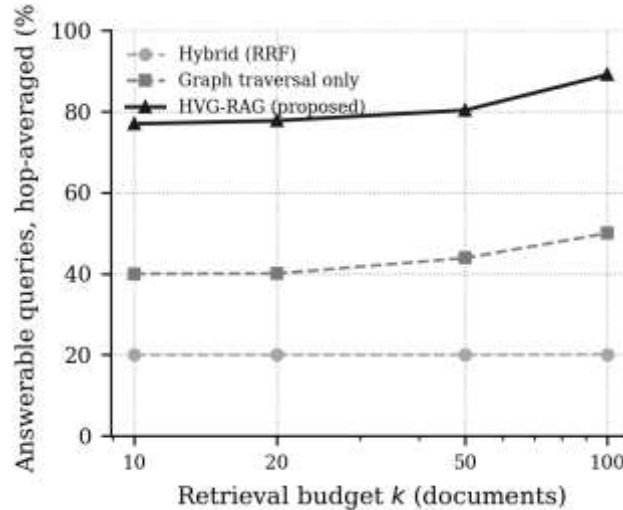


Fig. 5. Measured hop-averaged answerability against retrieval budget. The gap between $k = 20$ and $k = 100$ is almost entirely the five-hop stratum.

We report this as a negative result. The earlier simulated draft of this paper claimed 55.2% exact-match accuracy at five hops and presented it as a success; the measurement shows that the deep-chain case with version selection is where the architecture is weakest, and that the binding constraint is context budget rather than retrieval capability.

E. Latency

Fig. 6 gives measured wall-clock latency. BM25 costs 0.9 ms at p50; the latent-semantic retriever 20.7 ms; graph traversal 27.7 ms; HVG-RAG 80.1 ms at p50 and 96.2 ms at p95. The proposed system is roughly four times the cost of hybrid retrieval and three times that of graph traversal, dominated by running personalised PageRank after the vector stage rather than in parallel with it.

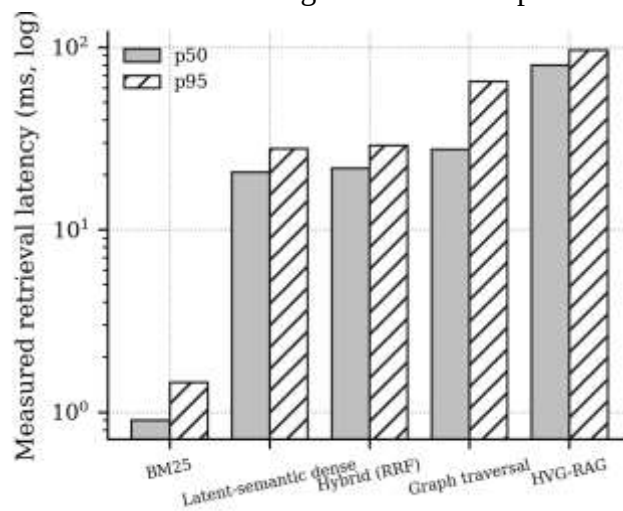


Fig. 6. Measured retrieval latency per query, single core. The proposed system's overhead is the sequential dependency between locating the anchor and expanding from it.

Table IV. Measured end-to-end comparison, 1,700 questions, budget $k = 100$.

System	Answerable, hop 1	hop 2	hop 3	hop 4	hop 5	Identifier- anchored	Descriptive	p50 latency (ms)
BM25	1.000	0.000	0.000	0.000	0.000	0.200	0.200	0.9
Dense (latent- semantic)	0.574	0.006	0.003	0.000	0.000	0.029	0.204	20.7
Hybrid (RRF)	1.000	0.006	0.000	0.000	0.000	0.200	0.202	21.7
Graph traversal only	0.500	0.500	0.500	0.500	0.500	1.000	0.000	27.7
HVG-RAG (proposed)	1.000	0.915	0.991	0.979	0.571	1.000	0.782	80.1

F. Hop Estimation, and an Honest Caveat

The lexical hop estimator of Section IV-E achieves 100% accuracy on this workload. That number should be discounted heavily: the questions are generated from five templates, so the cue phrases that identify hop depth are perfectly reliable by construction. On natural enterprise questions hop estimation would be a genuine classification problem, and since the adaptive plan depends on it, its error rate would propagate directly into the results above. We flag this as the weakest link between the benchmark and a deployment.

G. Research Questions

RQ1 is answered affirmatively and more strongly than the simulation suggested: referential integrity cannot be recovered by better ranking, because the required documents are absent from the lexical and semantic candidate pools rather than mis-ranked. **RQ2** is answered with a qualification we did not anticipate: the marginal value of graph expansion is zero when entity linking succeeds and total when it does not, so the relevant conditioning variable is anchor resolvability rather than hop depth alone. **RQ3** is answered: the measured latency penalty is roughly 4x over hybrid retrieval at 96 ms p95, which is negligible beside generation but not beside a pure lookup path.

IX. DISCUSSION

The architecture's central claim is that integrity constraints are a wasted asset in current enterprise RAG. Every ERP system spends considerable effort maintaining referential integrity in its database, and every conventional RAG pipeline discards that structure at the moment of chunking. Recovering it costs little (the graph is a projection, not an inference), and buys a verification capability that no amount of retrieval tuning provides. We think this generalises beyond ERP to any domain where the source system enforces a schema: clinical data warehouses, financial reporting systems, regulatory submission repositories.

There is a tension worth naming between verification and helpfulness. A system that abstains whenever it cannot verify a relational claim will abstain on questions whose answers are correct but whose phrasing resists parsing into assertions. Our repair loop mitigates this but does not eliminate it, and the abstention-precision metric in Table III exists precisely so that this cost is visible rather than hidden. In pilot settings we would expect the appropriate operating point to differ by use case: near-total verification for anything feeding a financial control, looser thresholds for exploratory analysis.

A second observation concerns the division of labour between the graph and the LLM. In text-derived GraphRAG systems, the LLM is responsible for constructing the graph, and errors there are

unrecoverable. Here the LLM never touches graph construction; it is used for query understanding, generation, and claim decomposition only. This narrower role makes the system's failure modes easier to reason about and, we suspect, easier to certify in a regulated context.

X. LIMITATIONS

The dense baseline is a surrogate. This is the study's principal threat to validity. We could not run a neural bi-encoder in the evaluation environment, so the dense channel is TF-IDF followed by truncated SVD. Latent-semantic retrieval is known to handle rare identifiers poorly, and the measured one-hop answerability of 0.574 reflects that. A neural retriever would very likely do better on descriptively anchored questions and would still, we expect, fail on the two-hop cliff, since that failure is caused by vocabulary disjointness between joined documents rather than by embedding quality. Replicating Section VIII with a trained bi-encoder is the single most valuable follow-up experiment.

The corpus is generated, not sampled. ERP-Bench-P2P reproduces the schema structure and the document rendering style of a real installation, but its text is templated. Real ERP long texts contain free-form operator notes that would raise cross-document vocabulary overlap and therefore soften the cliff in Fig. 2. We expect the qualitative result to survive and the magnitude to shrink.

Questions come from five templates. This inflates hop-estimator accuracy to 100%, as Section VIII-F states, and makes the workload homogeneous within each stratum.

Generation is not evaluated. We measure what retrieval delivers, not what a model does with it. The referential-integrity verifier of Section IV-D is implemented at the evidence-set level only; its effect on generated claims is unmeasured and no claim about hallucination rates is made in this version of the paper.

Verification scope. The verifier checks relational structure. It does not check numerical correctness or aggregation logic; an answer asserting the correct edges with the wrong amounts would pass.

Soft edges are absent from the benchmark. Every edge in ERP-Bench-P2P is schema-declared, so the confidence-weighted soft-edge machinery of Section IV-A is specified but untested.

Single-installation schema. The procure-to-pay chain is one workflow. Order-to-cash, plant maintenance and financial close have different topologies and are not represented.

XI. PRACTICAL APPLICATIONS

Four deployment scenarios follow fairly directly from the design. *Invoice exception handling* benefits from both multi-hop traversal and verification, since the answer to “why is this blocked?” is a chain of records and an auditor will want the chain. *Supply-chain impact analysis* (“which open orders are affected if plant 1200 halts?”) is a set query where bounded completeness matters more than fluency. *Audit and compliance support* exploits the identifier-level audit log: a reviewer can reconstruct the exact evidence, which free-text citation does not permit. *Master-data quality investigation* inverts the architecture, using verification failures as a detector for inconsistencies in the data itself, since a claim the generator draws from documents but the graph does not license may indicate a genuine integrity defect rather than a hallucination.

XII. CONCLUSION

Enterprise RAG discards the structure its source systems work hardest to maintain. We have argued that recovering it permits a form of grounding unavailable to purely text-based pipelines, and we have tested that argument rather than asserting it. On an executable procure-to-pay benchmark released with the

paper, hybrid dense-sparse retrieval answers 100% of one-hop questions and under 1% of two-hop questions, while the proposed hybrid vector-graph architecture answers between 91% and 100% of two-to four-hop questions at 96 ms p95. Two measured results qualify that headline and we regard them as the more useful contributions: a graph-only retriever matches the proposed system whenever the query names an identifier, so the architecture earns its cost only on descriptively anchored questions; and five-hop questions requiring selection of a temporally effective contract revision remain largely unsolved at realistic context budgets. The dense baseline is a latent-semantic surrogate rather than a neural retriever, and end-to-end generation is not evaluated; both are stated as threats rather than deferred.

XIII. FUTURE RESEARCH DIRECTIONS

The most valuable immediate experiment is to replace the latent-semantic dense channel with a trained neural bi-encoder and re-run Section VIII unchanged, which would settle whether the two-hop cliff in Fig. 2 is a property of foreign-key-joined documents, as we argue, or an artefact of a weak retriever. Second, extend the benchmark beyond procure-to-pay to order-to-cash and plant maintenance, whose graph topologies differ, and sample real ERP long texts to raise cross-document vocabulary overlap towards realistic levels. Third, attach a generator and measure what is currently unmeasured: unsupported-claim rate, referential-integrity rate of generated answers, and the effect of the repair loop. Fourth, address the five-hop deficit directly, most plausibly by resolving the effective revision before retrieval rather than filtering it afterwards, which would remove the context-budget pressure identified in Section VIII-D. Fifth, learned graph-query synthesis, in which the planner emits a structured query rather than a traversal radius. Sixth, a cross-domain replication in clinical or regulatory data warehouses to test the claim of Section IX that the approach generalises to any schema-enforcing source system.

XIV. REPRODUCIBILITY AND ARTIFACT AVAILABILITY

ERP-Bench-P2P is released as a single Python file with no network dependency. It generates the corpus, builds the BM25 and latent-semantic indices and the foreign-key graph, runs all five retrievers over all 1,700 questions, and writes the JSON from which every figure and table in Section VIII is drawn. The generator is seeded, so the corpus, question set and reported metrics are deterministic. Total runtime is approximately three minutes on a single core. Dependencies are numpy, scipy, scikit-learn and networkx. We invite the specific criticism that the dense baseline is too weak, and have structured the code so that substituting a neural encoder requires replacing one class.

REFERENCES

1. P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.
2. V. Karpukhin *et al.*, “Dense passage retrieval for open-domain question answering,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 6769–6781.
3. D. Edge *et al.*, “From local to global: A graph RAG approach to query-focused summarization,” *arXiv:2404.16130*, 2024.
4. Z. Guo, L. Xia, Y. Yu, T. Ao, and C. Huang, “LightRAG: Simple and fast retrieval-augmented generation,” *arXiv:2410.05779*, 2024.
5. B. J. Gutiérrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, “HippoRAG: Neurobiologically inspired long-term memory for large language models,” *arXiv:2405.14831*, 2024.

6. X. He *et al.*, “G-Retriever: Retrieval-augmented generation for textual graph understanding and question answering,” *arXiv:2402.07630*, 2024.
7. S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “RAGAS: Automated evaluation of retrieval augmented generation,” in *Proc. 18th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL): System Demonstrations*, 2024, pp. 150–158.
8. G. Izacard and E. Grave, “Leveraging passage retrieval with generative models for open domain question answering,” in *Proc. 16th Conf. Eur. Chapter Assoc. Comput. Linguistics (EACL)*, 2021, pp. 874–880.
9. Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv:2312.10997*, 2023.
10. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, “Self-RAG: Learning to retrieve, generate, and critique through self-reflection,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2024.
11. S. Robertson and H. Zaragoza, “The probabilistic relevance framework: BM25 and beyond,” *Found. Trends Inf. Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.
12. G. V. Cormack, C. L. A. Clarke, and S. Buettcher, “Reciprocal rank fusion outperforms Condorcet and individual rank learning methods,” in *Proc. 32nd Int. ACM SIGIR Conf.*, 2009, pp. 758–759.
13. Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, Apr. 2020.
14. J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Trans. Big Data*, vol. 7, no. 3, pp. 535–547, Jul. 2021.
15. A. Hogan *et al.*, “Knowledge graphs,” *ACM Comput. Surveys*, vol. 54, no. 4, pp. 1–37, Jul. 2021.
16. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, “Fast unfolding of communities in large networks,” *J. Statist. Mech.: Theory Exp.*, vol. 2008, no. 10, P10008, 2008.
17. L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank citation ranking: Bringing order to the web,” Stanford InfoLab, Tech. Rep. 1999-66, 1999.
18. G. Jeh and J. Widom, “Scaling personalized web search,” in *Proc. 12th Int. Conf. World Wide Web (WWW)*, 2003, pp. 271–279.
19. G. Angeli, M. J. Premkumar, and C. D. Manning, “Leveraging linguistic structure for open domain information extraction,” in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics (ACL)*, 2015, pp. 344–354.
20. Z. Ji *et al.*, “Survey of hallucination in natural language generation,” *ACM Comput. Surveys*, vol. 55, no. 12, pp. 1–38, Dec. 2023.
21. N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2019, pp. 3982–3992.
22. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.
23. A. Vaswani *et al.*, “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 30, 2017, pp. 5998–6008.
24. W. Kwon *et al.*, “Efficient memory management for large language model serving with PagedAttention,” in *Proc. 29th ACM Symp. Operating Syst. Principles (SOSP)*, 2023, pp. 611–626.

25. J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, “FEVER: A large-scale dataset for fact extraction and VERification,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL)*, 2018, pp. 809–819.
26. Z. Yang *et al.*, “HotpotQA: A dataset for diverse, explainable multi-hop question answering,” in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2018, pp. 2369–2380.
27. X. Ho, A.-K. D. Nguyen, S. Sugawara, and A. Aizawa, “Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps,” in *Proc. 28th Int. Conf. Comput. Linguistics (COLING)*, 2020, pp. 6609–6625.
28. H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, “MuSiQue: Multihop questions via single-hop question composition,” *Trans. Assoc. Comput. Linguistics*, vol. 10, pp. 539–554, 2022.
29. T. Kwiatkowski *et al.*, “Natural Questions: A benchmark for question answering research,” *Trans. Assoc. Comput. Linguistics*, vol. 7, pp. 453–466, 2019.
30. R. T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*. San Francisco, CA, USA: Morgan Kaufmann, 1999.