

Toward a Geometric and Compositional Knowledge Architecture for Organizational Knowledge

Andrew Miracle
Robin
me@andrewmiracle.com

Phyl Georgiou
Robin
phyl@withrobin.ai

Christine Okoth
Robin
christine@withrobin.ai

Abstract

Organizational knowledge systems need to move from storing and retrieving atomic knowledge to constructing contextual interpretations from that knowledge. Conventional document search, vector retrieval, and knowledge graphs each solve part of this problem and none solves it alone. Organizational knowledge is distributed across persistent domains; claims interpret evidence within those domains; and artifacts translate those interpretations for different audiences.

This paper proposes a conceptual architecture built from four layers: Signals (evidence with provenance), Domains (persistent semantic boundaries that classify Signals), Claims (propositions derived from Signals, carrying justifications and an explicit standing), and Dimensions (interpretable axes along which Claims can be positioned and compared). Claim standing is maintained by a justification-based dependency network in the tradition of truth maintenance systems, so that new or retracted evidence changes labels locally rather than triggering global recomputation. Dimensions follow Gärdenfors’ conceptual spaces in spirit but are specified as a constrained, versioned contract rather than as a learned embedding. The runtime separates a fast retrieval path that reads precomputed structure from a slow reasoning path that writes it.

The thesis is that expensive intelligence should write structure and cheap intelligence should read it. Nothing in this paper has been run. The evaluation plan states, for each component, the baseline it must beat, the metric, how the test set is built, the threshold that counts as success, and the result that would cause the component to be dropped, and it includes a cost ledger with a break-even condition for cheaper reads.

1 Thesis and Scope

This is an architecture proposal, not a report of results. It describes a design for a system for organizational knowledge, and it argues that the design is worth building by showing that each part has a concrete mechanism and a concrete test. Where a formal framework is invoked only for intuition, the text says so and keeps it short.

The thesis is a single claim: for an organizational knowledge base that is queried far more often than it is reasoned over, precomputing an interpretation layer (Claims with maintained standing, positioned on validated Dimensions) will improve retrieval quality and artifact quality relative to a system that keeps no such layer and instead retrieves Signals with a strong hybrid retriever and reasons over them per request when reasoning is needed, at a per-query cost that is lower, not higher, because the expensive reasoning is done once at write time and amortized across reads.

The thesis has a quality half and a cost half, each able to fail separately, both tested against the same alternative: no persisted Claims, hybrid retrieval over Signals for reads, per-request reasoning for artifacts. The quality half is wrong if that alternative matches the full system on either retrieval (Section 10.1, where the full system must beat a Signals-only hybrid arm significantly) or

artifacts (Section 10.2, where it must be significantly preferred to the slow path run per request with persistence disabled). The cost half is wrong if the total spend on slow-path writes, their automated maintenance, the human curation the structure requires, per-Signal ingest checks and fast-path reads does not fall below the alternative’s spend at the read/write ratio pilot workspaces exhibit (Section 10.4). In either case the system should be that alternative, not a knowledge model. The thesis is also wrong in a weaker way if the structure helps but goes stale faster than the slow path can maintain it; Section 10.4 states the staleness measurement that would show this.

Sections 2 and 3 define Signals, Domains, and Claims; Section 4 separates graph edges from geometry; Sections 5 and 6 define Dimensions; Sections 7 and 8 the runtime. Section 9 places the design against prior work and Section 10 gives the evaluation plan.

2 From Signals to Contextual Knowledge

Organizational knowledge should be stored at a finer granularity than documents. Entries are incoming material: conversations, meetings, emails, reports, database exports. The system extracts Signals from Entries: atomic pieces of knowledge, each carrying provenance back to its Entry and a semantic embedding. A Signal is atomic when it states one proposition that can be judged true or false without the rest of its Entry, with pronouns and references resolved. This is the proposition-level granularity of Chen et al. (2024) (Section 9).

Entries are inputs rather than a layer: an Entry yields Signals, and each Signal is classified into one or more Domains, the persistent semantic boundaries; a Domain might represent Climate Change, Kenya, Engineering Practice, Enterprise Sales, Customer Knowledge, or Organizational Relationships. The records below fix what each object holds. Provenance fields follow the minimum that PROV-DM (Moreau and Missier, 2013) asks of a derived entity: who or what produced it, when, and from what.

- **Entry** = (entry_id, source, author, received_at, content, withdrawn). Never edited. Its only later operations are withdrawal and restoration, by a person or by the source system.
- **Signal** = (signal_id, proposition, entry_id, span, extractor_id, extracted_at, observed_at, embedding, domains, supersedes). Produced by extractor_id (a model and version) at extracted_at from span of entry_id; observed_at is when the Entry says the fact held, or received_at if it does not say. Exactly one Entry, one or more Domains, at most one supersedes link to an earlier Signal from the same source, set at extraction when the source marks a record as replacing an earlier one or the extractor finds the new Signal restating one of that source’s earlier Signals with a later value; a person can remove a wrong link, restoring the earlier Signal. Invariant: the proposition is entailed by its span alone and is never edited. Operations: extract, classify, reclassify, supersede.
- **Domain** = (domain_id, name, description, θ_D , A_D , version, author, created_at, derived_from, status). derived_from names the parent Domain for each Domain created by a split, and the predecessor version for each revision, and is empty for a Domain created new. Classification logic: the Domain classifier is a decision-model call (Section 7) whose state is the Signal (on the read path, the query, in the call shared with the Router) and whose questions ask, for each Domain description, whether the Signal falls under it; a Signal joins D when the returned probability is at least θ_D , and joins its single highest-scoring Domain when no threshold is met, so no Signal is without a Domain. Creation, description revision, splitting and retirement are human actions recorded with author and time. Creation and revision (which bumps the

version) score every Signal against that Domain only, and a split is two creations followed by a retirement; the re-scores commit through slow-path batches, Claims whose home sets change (Section 3.1) are re-indexed, and none is re-derived. Retirement removes the Domain from every Signal and home set, reassigns Signals left without a Domain by the highest-score rule, and parks its Dimensions.

This classifier is the semantic routing layer; because Domain scope is a hard filter on the fast path, its precision and recall matter downstream, and Section 10.6 measures them.

Suppose the system knows four things: Foundation X funds climate adaptation programs; Consortium Y operates within several Kenyan counties; the organization has an established relationship with Consortium Y; and county governments control a relevant implementation mechanism. None alone implies a strategy. Suppose a user asks how the organization could reach Foundation X’s county-level climate program. The slow path retrieves across the Domains in scope and, from the four Signals together, derives: partnering with Consortium Y may provide the strongest route for accessing and implementing Foundation X’s county-level climate program. The resulting Claim is homed in every Domain its Signals came from and is stored, so later requests can read it without deriving it again.

Each existing approach fails this example at a different step. Suppose the four Signals sit in four Entries: a funder announcement, a partner profile, a relationship note, and a policy memo. Document search ranks Entries by term overlap with the request, whose content terms are Foundation X, county, climate and program: the announcement matches three, the memo and the profile one each (county), and the relationship note none, so the note, which carries the route, ranks last, and no document states the route. Vector retrieval ranks Signals by similarity, and “the organization has an established relationship with Consortium Y” is not similar to a question about Foundation X; at best it returns four true statements and nothing about what they imply together. It fails a second case too: asked whether the program is still funded, it returns “Foundation X committed \$50 million” and “Foundation X cancelled the program” at near-equal similarity (Section 5.3), and nothing says which holds. A knowledge graph can join the four by a path query, if someone has modelled those relations and written it; the judgement that the path may be the strongest route is not an edge it holds with a revisable standing. In a second case, an edge “a person P leads the Kenya program” stays after a memo says P has moved on, until someone edits it, and nothing records which answers relied on it. Combining them, as hybrid or graph-augmented retrieval does, can put all four Signals in one context for a model reasoning per request. The combination fails at keeping the interpretation, not at finding the evidence: the proposition is re-derived on every request, possibly differently, never checked against later evidence, and when Consortium Y leaves the counties nothing records that earlier answers rested on it. That combination is the alternative Section 10 tests.

That proposition is not another Signal. It is an interpretation produced by combining evidence. Operationally, a response retrieves when every proposition it presents is a stored object or is entailed by a single stored object, and it constructs an interpretation when it presents a proposition that no single object it cites entails. An interpretation is kept when the system stores it with its justification, and transient otherwise. Per-request reasoning, the strongest alternative, constructs transient interpretations; this design keeps them. The partnership proposition is an interpretation by this test, since none of the four Signals entails it. This is the role of the Claim.

3 Claims as an Interpretation Layer

3.1 The Claim record

A Claim is a record

$$C = (\text{id}, \text{proposition}, \text{embedding}, J, \text{context}, \text{standing}, \text{history}, \text{override}, \text{lifted}, H)$$

where the proposition is a natural-language statement and the embedding its vector; history is the time-stamped log of standing changes with their causes, which the rules of Section 3.3 read; override records a user rejection or a supersedes edge, if any (Section 3.4); lifted marks a lifted Claim; J is a non-empty set of justifications, each a set of antecedent Signals and Claims with a derivation record (which model or person produced it, when, and for which request or, for a supports verdict or precondition migration, which write); context holds the unevidenced preconditions (Section 3.2); standing is one of the five states of Section 3.3; and H , the Claim’s home, is the union over its justifications of its Signal antecedents’ Domains and its Claim antecedents’ home sets, recomputed when J or an antecedent’s Domains change and well defined because the justification graph is acyclic (Section 3.4). Assertion time is in the derivation record and observation times are on the Signals in J .

The boundary between Signal and Claim is decided by origin and entailment. An object is a Signal if it was extracted from one Entry and its proposition is entailed by that Entry’s span alone. An object is a Claim if the slow path or a user wrote it. A derivation must produce a proposition that no single antecedent entails; one that only restates an antecedent writes nothing and cites it instead. Both entailment conditions are checked by the classifier of Section 3.4, at one call per extracted Signal and per derivation antecedent, charged to ingest and $W(m)$ (Section 10.4); a misjudged object surfaces through the contradiction check or a user’s rejection. A supports verdict (Section 3.4) is corroboration, not derivation, so the rule does not apply to it. There are two exceptions, both restatements by construction: the lifted Claim of Section 3.4, which is never retrievable, and a user-entered Claim, whose justification is the Signal extracted from the Entry that records the user’s statement, plus any objects the user cites.

Consider a client saying, “Users should be able to see everything we sell and quickly narrow down the options”, and a Signal in an Engineering Domain that large catalogs commonly require faceted filtering, product grids and pagination. The client’s sentence is a Signal. “The catalog likely requires faceted filtering” is a Claim, because neither the client’s Signal nor the Signal about large catalogs entails it alone.

The same Claims support very different artifacts: an engineering specification lists a filter panel, product grid and pagination, while a stakeholder document says customers can browse, narrow and select products efficiently. The interpretation is the same and only the renderer changes, so artifact generation is not the reasoning layer; artifacts render the Claim model.

3.2 Where Claims live, and what context holds

A Claim is stored once and indexed under every Domain in its home set H . Because later requests may read it under different assumptions, its preconditions cannot stay implicit in the request that produced it, and they are handled in two ways, depending on whether evidence for them exists.

An evidenced precondition is one that some Signal or Claim in the workspace asserts. When the slow path derives a Claim under an assumption, it is required to search for a supported Signal or Claim asserting that assumption; if one is found, that node is added to the antecedent set

of every justification of the new Claim. The assumption then lives in J , not in context, and the dependency network of Section 3.4 retracts the Claim automatically when that evidence is withdrawn or superseded (Section 7 follows one such precondition).

An unevidenced precondition is an assumption for which no evidence was found. Only these are stored in context. Two mechanisms read the field. The renderer must print every unevidenced precondition beside any Claim it cites, in every audience register, and Section 10.2 has a rater check for this. And on every slow-path run that touches the Claim, the search for evidence is repeated; a precondition that acquires evidence moves from context into J . This migration adds the supported node to every justification of an existing Claim, passes the cycle check of Section 3.4 first, records the run and time that added it in each justification’s derivation record, and thereafter the Claim depends on that node like any antecedent; a migration the check rejects leaves the precondition in context. Unevidenced preconditions are the assumptions an assumption-based truth maintenance system would label nodes with, the placeholder for that extension (Section 9).

3.3 Standing

Standing is the Claim’s status in the system’s current view of the evidence, one of five states: proposed, accepted, contested, retracted, and superseded. It is not a probability (Section 9 says why factor graphs were not adopted). The states and their transitions are:

- **proposed:** the initial state of any Claim written by the slow path or entered by a user. A proposed Claim is retrievable, penalized in ranking by the standing term μ of Section 6 while Section 10.1 keeps that term, and marked as unreviewed in artifacts; lifted Claims (Section 3.4) are the exception and are never retrievable.
- **proposed to accepted:** a user accepts the Claim, or the network applies the automatic acceptance rule: at least one justification has antecedent Signals from at least two distinct Entries, all antecedents are supported, the contradiction check found no conflicting Claim, and the Claim has never been retracted. The rule is evaluated when the check completes and whenever propagation re-evaluates the Claim. Because contest does not travel along justifications (Section 3.4), a Claim resting on a contested Claim can still be accepted. Workspaces may disable the automatic rule and require a person for every acceptance.
- **accepted or proposed to contested:** the dependency network records a conflicts_ with edge between this Claim and another Claim whose standing is neither retracted nor superseded. Contested Claims remain retrievable; the reranker penalizes them, and any artifact citing one must surface the conflict alongside it.
- **contested to accepted, or contested to proposed:** the conflicting partner becomes retracted or superseded (by propagation, by a user rejecting it, or by a supersedes edge), or a user marks the conflict spurious, deleting the edge; there is no third option. The Claim returns to accepted if it had been accepted before the conflict and to proposed otherwise.
- **any state to retracted:** no justification remains supported (every justification has an unsupported antecedent: one retracted by propagation, overridden by rejection or supersession, or withdrawn at source; Section 3.4), or a user rejects the Claim. Retracted Claims are excluded from retrieval and kept for audit.
- **retracted to proposed:** a Claim retracted by propagation regains support because a withdrawn Entry is restored or a superseding Signal is itself withdrawn; since the automatic

rule excludes Claims ever retracted, only a person can accept it again. A Claim retracted by a user does not re-enter on regained support.

- **any state to superseded:** the slow path or a user writes a revised Claim with a `supersedes` edge to this one. The superseded Claim is excluded from retrieval by default and remains reachable from its successor; deleting a wrong edge returns it to the standing its support and history give.

Users move Claims by acceptance, rejection, supersession and conflict resolution; the dependency network moves them when antecedent support changes and when the automatic rule fires; the slow path creates Claims and supersession edges but never sets standing to accepted.

3.4 Dependency tracking

The system maintains standing with a justification-based dependency network, adapting the justification-based truth maintenance approach of Doyle (1979). The mechanism is small enough to state completely.

Nodes are Signals and Claims. A Signal is supported unless its source Entry has been withdrawn or a later Signal from the same source supersedes it (Section 2). A Claim is supported when it carries no override and at least one of its justifications has all antecedents supported. A `conflicts_with` edge is a symmetric relation between two Claims asserted by the contradiction check below; each edge records which branch of the check wrote it. Two overrides sit above the support computation: a `supersedes` edge sets a Claim’s standing to superseded from any state, and a user rejection sets it to retracted, taking precedence if both apply; a person can delete a wrong `supersedes` edge, which removes that override. Propagation never changes an override, and an overridden Claim counts as unsupported, so writing an override starts propagation from that Claim; this is how rejecting an antecedent retracts its dependents. Deleting a `conflicts_with` edge is an edit, not an override. For a Claim with no override, standing is a function of support and conflict: an unsupported Claim is retracted; a supported Claim with a conflicting partner whose standing is neither retracted nor superseded is contested; a supported Claim with no such partner is accepted if it has been accepted and not since retracted, and proposed otherwise.

The update rule is propagation along justification edges, in three steps. When a Signal or Claim changes support status, the network first collects its forward closure: every Claim that lists it as an antecedent, every Claim that lists one of those, and so on. It then re-evaluates each Claim in the closure once, in topological order, so that each is evaluated after all of its antecedents, and finally re-evaluates once the conflict partners of every Claim in the closure, the changed node included, whose standing changed; a partner’s support is unchanged, only its standing, so it starts no further propagation. Because the justifications are monotone (a Claim is justified only by the presence of supported antecedents, never by the absence of something) and the graph is acyclic, this single pass computes the unique labelling in which every supported Claim has well-founded support, and support does not depend on the order in which changes arrive. Standing can depend on order, by design, because its rules read a Claim’s history. A general justification-based system needs two-phase retraction, marking out everything that rested on a withdrawn node before searching for alternative support, or circular support can keep a node in (Forbus and de Kleer, 1993); acyclicity makes one pass enough here. Doyle’s non-monotonic justifications, which support a belief on condition that another node is out, are excluded, so “no competing bid has been submitted” must be justified by a Signal that asserts it.

Support must also be well-founded (Forbus and de Kleer, 1993): a pair of Claims that justify each other with no Signal beneath them must not count as supported. Cycles cannot arise from

derivation, because a derived Claim’s antecedents exist before it does. Three paths add antecedents to an existing Claim: the supports verdict below, re-derivation that re-supports a Claim with new evidence, and precondition migration (Section 3.2). Before any of them writes, the network rejects the write if the Claim is an ancestor of a new antecedent, by a reachability search linear in that antecedent’s ancestors and charged to the write. With that check, every supported Claim has a finite justification chain ending in Signals.

Locality is bounded by that forward closure plus the conflict partners of its Claims, not by a Claim’s direct dependents: relabelling visits the closure’s Claims, its justification edges and those partners once each, with no model calls. The closure can be wide: a Signal cited by most Claims in a Domain brings them and their descendants in, and in the worst case it is every Claim in the workspace; the design only makes that case a graph traversal. Contest does not travel along the closure: a `conflicts_with` edge changes the standing of the two Claims it joins and no others, deliberately, since a spreading contest would mark everything downstream of one disagreement.

Propagation changes labels and regenerates no text, which keeps it cheap. When a Claim becomes retracted or contested, the slow path is scheduled to re-derive it only if the Claim was retrieved in the last thirty days, which the fast path’s retrieval marks (Section 7) record. Re-derivation means running derivation again for the same question the Claim answered against the evidence now present: if the proposition that results differs, it is written as a new Claim with a supersedes edge to the old one; if the same proposition is re-supported by different evidence, the new justification is added to the old Claim; otherwise nothing is written. This is where a wide change costs model calls, at most one derivation per relabelled Claim that became retracted or contested and was recently retrieved, so its worst case is also the closure and its partners; Section 10.4 charges it as automated maintenance.

Writes to a Domain are processed in slow-path batches flushed at an interval Δ ($\Delta = 5$ minutes initially). Within a batch, a new Claim is projected onto at most K active Dimensions of its home Domains (Section 5.5) before its contradiction check, so the check sees positions; that adds at most K projector calls per Claim, on the frontier projector until the small one qualifies (Section 5.3). A contradiction is detected no later than Δ plus the batch’s processing time after the contradicting object is written. A batch commits atomically: its Claims, projections, edges and standing changes become visible to readers together (Section 7).

The contradiction check runs whenever a Signal or Claim is written to a Domain, with a candidate step that differs by input type. When the new object is a Claim, candidates are the k nearest existing Claims in the same Domain by embedding similarity ($k = 20$ initially), restricted to those sharing a Dimension d on which both have relevance $r_d \geq \tau_r$ and positions differing by $\delta_d > \tau_\delta$ ($\tau_r = \tau_\delta = 0.5$ initially): the Section 5.3 signature of high topical similarity and opposite position. If fewer than five candidates survive, the filter is relaxed to similarity alone so that Claims with no Dimension coverage are not exempt. When the new object is a Signal, which is not projected (Section 5.3), candidates are the k nearest Claims by similarity above a floor τ_s , set permissively at the lowest decile of Signal-to-nearest-Claim similarities in the Domain, below which a Signal costs no classifier calls. In both cases a classifier then labels each (new, candidate) pair as supports, contradicts, or unrelated: by default a small natural-language-inference model (Bowman, Angeli, Potts, and Manning, 2015; Williams, Nangia, and Bowman, 2018), on the task de Marneffe, Rafferty, and Manning (2008) found harder than entailment and on which, contradictions being rare, precision matters most (Section 9). A contradicts verdict writes a `conflicts_with` edge tagged with the branch that produced it; a supports verdict adds a new justification to the Claim whose single antecedent is the new object, subject to the cycle check above. Adding a justification can only add support. Precondition migration is the one write that narrows support, deliberately: it makes explicit a dependence the derivation already assumed.

Two kinds of conflict are excluded: Signals are not checked against Signals, since disagreement is

recorded only where an interpretation rests on it; and a Claim written after a Signal that contradicts it is checked only against Claims, the slow path’s reading of the Domain’s Signals being the only guard in that direction. No rule picks a side in a recorded conflict: a contest ends only when a person rejects a partner or marks the conflict spurious, or when a partner loses support or is superseded, by a person or by a re-derivation from new evidence (Section 3.3).

Because `conflicts_with` joins two Claims, a contradicting Signal is first lifted into a proposed Claim that restates its proposition and is justified by that Signal alone, and the edge joins the lifted Claim to the one it contradicts. The lifted Claim is supported as long as its Signal is, which keeps the contradicted Claim contested; it is not projected, triggers no check and is not indexed for retrieval, since the Signal is what a reader should see. If a later slow-path run derives the same proposition as an ordinary Claim, that Claim takes over the edge. A Claim written with no supported antecedent is retracted on arrival and contests nothing. The cost is at most k classifier calls plus one lifting call per write, carried as its own line in Section 10.4.

Suppose the partnership Claim of Section 2 has been accepted, and a partner update from a different source yields a Signal stating that Consortium Y has withdrawn from the counties. The check finds the partnership Claim among the new Signal’s nearest Claims; if the classifier labels the pair contradicts, the Signal is lifted, a `conflicts_with` edge is written, and the partnership Claim becomes contested, so any artifact citing it must show the conflict. A Claim about Foundation X’s funding that never cited the Consortium Y Signal is not touched. The same-source case, where the update supersedes the Signal and propagation retracts the Claim, is followed in Section 7.

4 Explicit Graphs and Geometric Relationships

The architecture contains many relationships naturally represented as graph edges: a Signal is extracted from an Entry; a Person authored an Entry; a Signal is classified into a Domain; a Signal or Claim justifies a Claim; an Artifact cites a Signal or Claim; a Claim conflicts with or supersedes another Claim. These edges answer who said something, where it originated, and what supports it.

Other relationships cannot be named so precisely: two Claims may be similar without either supporting the other, may concern the same subject and take opposite positions, or may sit one between two others on some axis. These are geometric. In Gärdenfors’ conceptual spaces, entities are positioned on interpretable quality dimensions and similarity is related to distance, an intermediate level between symbolic and connectionist representation (Gärdenfors, 2004). An embedding also provides geometry, but its coordinates are opaque; named axes, such as implementation feasibility, say along which axis two Claims are close or far. Graphs encode what the system can name, and geometry encodes relative position.

One terminological collision must be resolved. Gärdenfors (2000) uses “domain” for a set of integral quality dimensions perceived together, such as hue, saturation and brightness. The system’s Domain is a topical routing boundary (Climate, Kenya), a different object; Gärdenfors’ notion is written “quality domain” here. Its nearest analogue would be a group of integral Dimensions within a Domain, which this design does not model: Dimensions are treated as separable, and Section 5.4’s redundancy check is the only place their interaction is examined.

5 Dimensions as Validated Organizational Structure

A Dimension is not a tag: “funding” is a topic, while “funding availability” can become a Dimension only if knowledge objects can meaningfully be compared on it.

5.1 The Dimension contract

A Dimension is a specification with seven fields,

$$d = (V_d, \Pi_d, \rho_d, \nu_d, P_d, \text{prov}_d, v_d)$$

each drawn from or checked against the registry of Section 5.2:

- V_d , the value space and its range: a bounded scalar in $[0, 1]$ or an ordered scale of named levels. Unordered categories are not admitted, since an axis whose values have no order is a facet, not a Dimension.
- Π_d , the projection system: a projector model and a prompt filled with the Dimension’s name, definition text and prototypes, returning a value in V_d and a relevance in $[0, 1]$ (Section 5.3).
- ρ_d , the comparison function on V_d : absolute difference for scalars, rank distance for ordered scales.
- ν_d , the normalizer that turns ρ_d into $\delta_d \in [0, 1]$, for instance division by the range or by the number of levels minus one.
- P_d , two to five prototypes: reference propositions with assigned values, shown to the projector as anchors. They calibrate projection; they are not category prototypes in Gärdenfors’ sense, and no Claim is classified by its nearest prototype.
- prov_d : who proposed it (model proposal or disentanglement with the model identifier, or a named person for a human-authored Dimension), when, the Claim-pair sample it was proposed from, its validation results with the projectors used (Section 5.4), and what activated it, the Useful screen or a named user’s pin.
- v_d , an integer version (Section 5.5). Outside the version, each Dimension also has an identifier, an owning Domain and a lifecycle status (candidate, validated, active, parked or retired; Section 5.4); its name and definition text are part of Π_d .

Implementation feasibility, used below, has $V_d = (\text{blocked}, \text{constrained}, \text{feasible}, \text{ready})$, rank distance as ρ_d , and division by three as ν_d . Models write candidates, the Useful screen or a user activates them, and people decide replacements and retirements. “Interpretable” means three checkable things: the axis has a name and a definition, its values are assigned by a stated procedure (the projection system), and the assignments are checked for consistency by the Measurable and Stable criteria of Section 5.4.

The contract keeps some properties of conceptual spaces and drops others. It keeps metric structure on each Dimension, through ρ_d and δ_d ; separability, since Dimensions are compared one at a time and their terms summed with weights in Section 6, which is city-block in form, the combination Gärdenfors (2000) prescribes for separable dimensions, although the summed terms are relevance-weighted similarities rather than distances, so the score is not itself a metric; and betweenness on a single ordered Dimension, where y lies between x and z on d when $\delta_d(x, y) + \delta_d(y, z) = \delta_d(x, z)$. It drops integral grouping (Section 4); betweenness across several Dimensions, which would need a joint metric the design does not define; convex regions, because the system never represents a concept as a region, so convexity has nothing to apply to; and prototype-based categorization, for the reason given under P_d . What it keeps is what reranking and the contradiction filter use.

Three arguments favour a specified axis over a learned embedding coordinate here. Auditability: every coordinate has a definition, a projector and a version, so a ranking or contradiction candidate

can be explained by naming the axis and checked by re-projecting. Stability: retraining an embedding moves every coordinate at once and records nothing per axis, while a Dimension changes one axis at a time, under a version (Section 5.5). Authoring cost is the embedding’s real advantage, since it costs no human time per axis; the contract bounds that cost, with models proposing and validating and people acting only on replacements, pins and retirements, which Section 10.4 charges and Section 10.5 compares with human authoring.

Alshaikh, Bouraoui, and Schockaert (2020) observe that interpretable features often pertain only to subsets of entities, and partition an embedding space accordingly; the system mirrors that with Domain-specific Dimensions (a Climate Domain might carry funding availability and policy support, an Engineering Domain reliability and reversibility), and their method is a candidate generator in Section 5.4.

5.2 A constrained Dimension specification

Models may not invent executable mathematics. Each field of Section 5.1 references an implementation in a controlled registry of value spaces, projection systems, metrics and normalizers; a typed schema constrains model-generated specifications, and a compatibility validator rejects mismatched components (rank distance on a scalar value space). A model composes approved primitives; it does not generate formulas. The normalizer maps ρ_d to a standardized dissimilarity

$$\delta_d(x, y) \in [0, 1]$$

that is 0 when two positions are indistinguishable and 1 when they are maximally different on d . This lets different native representations coexist while still permitting comparison at the reranking layer.

5.3 Four quantities

There are four independent quantities for a Claim x and Dimension d . The value $\pi_d(x) \in V_d$ is where x lies on d . The relevance $r_d(x) \in [0, 1]$ is how strongly x concerns d at all; the relevance threshold $\tau_r = 0.5$ is the single cut used wherever the paper asks whether an object is relevant to a Dimension. The distance $\delta_d(\pi_d(x), \pi_d(y))$ is how different two positions are. The salience $w(d)$, a fitted coefficient (Section 6), is how much d matters in a Domain.

The two funding statements of Section 2, a commitment and a cancellation, are both highly relevant to funding availability, likely close in embedding space, and far apart on that axis. This is why the two geometries stay distinct, and, once both are Claims with projections, it is the signature the Claim branch of the contradiction check (Section 3.4) exploits.

Projection produces $(\pi_d(x), r_d(x))$ for a Claim. Signals are not projected: most are never cited by a Claim, and the reranker scores Claims, so projecting them would multiply cost for no read-side benefit. Each call gives the projector one Dimension’s definition and prototypes and one proposition, and returns a value in V_d and a relevance, which keeps the per-Dimension agreement of Section 5.4 unconfounded. Queries are projected with the same convention, but only by the small projector (Section 6). Chatterjee, Gajbhiye, and Schockaert (2023) found that language models recover some conceptual-space rankings, unevenly across properties, and that much smaller fine-tuned models can match a much larger one. Hence the plan: bootstrap projections with a strong model, measure agreement (Section 5.4), and move projection to a small model fine-tuned on the bootstrap (Wang, Liu, Xu, Zhu, and Zeng, 2021) once agreement is acceptable. Their properties are general commonsense ones, not organization-specific axes, so Section 10.5 measures whether this holds here.

5.4 Discovery and validation

Dimensions are hypotheses about useful axes of organizational knowledge, not permanent declarations. A candidate Dimension is first validated, then activated if it proves useful, and eventually parked or retired. The active set of a Domain D is written A_D .

Candidates come from two sources. The first is a model proposal step, run per Domain once per evaluation cycle (initially seven days): the model is given the Domain description, the definitions of its active Dimensions, and a sample of Claim pairs close in embedding space that the reranker ranked far apart for the same queries, the residual the current geometry does not explain, and returns zero or more candidate specifications per Section 5.2. The second, compared against the first in evaluation, is an Alshaikh-style disentanglement of the Domain’s Claim embeddings, whose directions a model then names and defines as specifications.

Validation runs against a sample S of n Claims from the Domain ($n = 100$ where the Domain has that many; minimum 30). Rank agreement below is written r_s (Spearman’s coefficient) to keep it apart from the Dimension’s native comparison ρ_d . The criteria are operational tests, with initial thresholds that are settings to be tuned, not findings:

- **Measurable.** Two independent projections of S , produced with different prompt phrasings of the same definition or with the two projector models named under Stable, agree: $r_s \geq 0.7$. - **Covered.** At least 20% of S has relevance $r_d \geq \tau_r$; otherwise the Dimension is a tag, not an axis.
- **Distinct.** The candidate’s definition embedding has cosine similarity below 0.9 with every existing Dimension definition in the Domain. This catches renames before spending projection budget.
- **Non-redundant.** On Claims relevant to both, $|r_s(\pi_d, \pi_{d'})| < 0.8$ against every active Dimension d' in the Domain. A candidate that fails this against exactly one d' is a candidate replacement for d' and is queued for human review rather than rejected.
- **Stable.** The sample is re-projected with a different projector model at validation time and again after thirty days or after the Domain’s Claim count has grown by 20%, whichever comes first. Agreement with the original projection meets the Measurable thresholds both times. Stability is re-checked on every version bump that makes projections stale (Section 5.5).

A candidate passing these becomes validated and is projected only for the Claims the Domain’s validation queries retrieve; Domain-wide projection happens on activation, which the cost formula of Section 5.5 counts. It becomes active, entering A_D , when the Useful screen passes: the reranker of Section 6 is refit with five-fold cross-validation on the validation split, and the candidate passes when $\beta_{D,d}$ is non-zero and out-of-fold nDCG@10 is higher with it than without it, each in a majority of folds; a tie is a fail. This is a cheap screen, not a test; Section 10.1 tests whether active Dimensions help, on a held-out split the screen never sees. A Domain below the fitting minimum of Section 6 cannot run the screen, and its validated Dimensions stay inactive until it can. A Dimension pinned by a user is active regardless. A Dimension that fails the screen in two consecutive evaluation cycles is parked: kept, not projected, not used. Retirement is a human action.

Dimensions are discovered from Claims, and standing depends on positions through the contradiction filter, but the dependence is stratified in time: discovery reads Claim text, embeddings and reranker residuals as of the start of a cycle and never reads standing; standing reads positions only through the candidate filter, which writes none; and a newly activated Dimension affects only later checks. The active sets are fixed before each batch, and standing within it is a function of them.

5.5 Projection cost and invalidation

Projection is the per-Claim write cost the Dimension layer adds. For a Domain D with Claim set C_D and active Dimension set A_D , the projection cost is

$$\text{cost}(D) = \sum_{d \in A_D} |\{c \in C_D : \hat{r}_d(c) \geq \tau_g\}| \cdot c_\pi$$

where c_π is the cost of one projection call and $\hat{r}_d(c)$, the cosine similarity between Claim and Dimension-definition embeddings, gates projection at a threshold τ_g , applied before any model is called and set permissively (initially at the 30th percentile of similarities in the Domain) to remove only Claims that plainly do not concern the Dimension. Without the gate the cost is $|C_D| \cdot |A_D| \cdot c_\pi$. Three controls bound it: the gate; a cap on the active set of $|A_D| \leq K$ per Domain ($K = 8$ initially), where a Dimension passing the Useful screen at the cap displaces the active one with the smallest mean out-of-fold $\beta_{D,d}$ (ties park the most recently activated), and a Claim homed in several Domains is projected on at most K Dimensions of their active sets, those with the largest mean $\beta_{D,d}$, so cost grows linearly in Claims and the sum above is an upper bound; and a lower c_π , from moving projection to a small fine-tuned projector once it meets the Measurable thresholds against the frontier bootstrap (Section 10.5). Projections are computed on the slow path in batches, when Claims are written (before their contradiction check, per Section 3.4) or when a Dimension is activated, never on the fast path.

The contradiction check of Section 3.4 also runs on every ingested Signal at up to k classifier calls each, and Signals outnumber Claims. Per Claim-equivalent, the two costs are $K \cdot c_\pi$ for projection and $R_{SC} \cdot k \cdot c_{\text{nli}}$ for ingest checks, where R_{SC} is the Signal-to-Claim ratio and c_{nli} the cost of one classifier call. Which dominates total write spend depends on R_{SC} and on c_{nli}/c_π , neither of which this paper assumes; both are measured in Section 10.4, and the ledger there carries the two as separate lines. The similarity floor τ_s of Section 3.4 is the control on the ingest side.

Every projection is stored as a record (claim_id, dimension_id, dimension_version, projector_id, value, relevance, projected_at). A Dimension’s version v_d increments whenever any field other than prov_d changes, so recording a validation result cannot trigger another check. A change to anything that enters the projection prompt (value space, projection system, definition text, prototypes) makes projections under earlier versions stale; a change to the metric or normalizer leaves stored values valid, since both act at comparison time, but still bumps the version, so every comparison is traceable and $\beta_{D,d}$ is refit at the next cycle. Stale projections are excluded from reranking (the term drops out for that Claim) and from the contradiction check’s position filter. A retrieved stale Claim is queued for re-projection in the next slow-path batch, and a background job backfills the rest by access frequency; a retired Dimension’s projections are kept for audit and never read. A version bump thus costs up to $|C_D| \cdot c_\pi$ over time, which is why the Stable criterion exists.

6 Dimensions and Domains

Domains provide broad semantic routing and Dimensions interpretable structure inside Domains. The architecture routes first and positions second: a query or Signal is routed to Domains by the Domain classifier, the decision-model call of Section 2, and Claims within a Domain are positioned on that Domain’s Dimensions by projection.

The reranking score is where salience becomes concrete. For a candidate Claim c retrieved for query q , with hybrid base score $b(c, q)$ from the fusion-and-cross-encoder stage, define the per-Dimension feature

$$\phi_d(c, q) = r_d(c) \cdot (1 - \delta_d(\pi_d(c), \pi_d(q)))$$

where $\pi_d(q)$ is the query’s target position on d , obtained by projecting the query with the same one-Dimension-per-call convention as Claims but always with the small projector: a Dimension whose small projector has not met the Measurable thresholds is treated as expressing no query position, so query projection costs up to $|A_D|$ small calls per Domain in scope and never calls a large model, issued in parallel, with latency of one call. The query expresses no position on d when the projector’s relevance output for it is below the relevance threshold, $r_d(q) < \tau_r$; in that case the factor $(1 - \delta_d)$ is replaced by 1 and ϕ_d reduces to the Claim’s relevance. The Dimension-aware score is

$$\text{score}(c, q) = b(c, q) + \sum_{d \in A(c)} w_q(d) \phi_d(c, q) - \mu \cdot [\text{standing}(c) \in \{\text{contested}, \text{proposed}\}]$$

where $A(c)$ is the set of at most K Dimensions, drawn from the active sets of c ’s home Domains, on which c is projected (Section 5.5) and has a non-stale projection.

The salience is fitted per Domain and Dimension. For each Domain D and $d \in A_D$, $\beta_{D,d}$ is fit on the validation split by pairwise learning-to-rank (Burgess et al., 2005) over the features ϕ_d with graded relevance labels, constrained non-negative (closeness on a Dimension should never count against a Claim) and L_1 -penalized (Tibshirani, 1996) so that unhelpful Dimensions receive exactly zero. The weight $w_q(d)$ is $\beta_{D,d}$ for the Domain in the query’s scope that is a home Domain of c with d active, taking the maximum when several qualify. The penalty μ on unsettled standing is fit in the same procedure, or alone, as Section 10.1 uses it. A Domain needs at least 30 validation queries whose scope includes it; below that an L_1 fit would zero Dimensions for want of data, so the Domain keeps its Dimensions inactive and is reported as unfit. The fitted $\beta_{D,d}$ is what the Useful screen reads; there is no global scale to tune. Coefficients are refit once per evaluation cycle, when the Useful screen runs; between refits they are constants, and a refit takes effect at a batch commit.

7 Fast Search and Slow Reasoning

The fast path is search. It optimizes for latency and retrieval quality. Its inputs are precomputed state: Domain assignments, Signal and Claim embeddings, Claim Dimension positions, fitted Dimension coefficients, BM25 indexes, provenance edges, and Claim standing. Its model calls in the ordinary case are the query embedding, the cross-encoder reranker, and the query projection of Section 6, all small; the one exception is the escalation described next. A typical fast path scopes the query to Domains, retrieves candidates by BM25 (Robertson and Zaragoza, 2009) and dense retrieval (Karpukhin et al., 2020), reranks them with the cross-encoder (Lin, Nogueira, and Yates, 2021; Nogueira and Cho, 2019) and then with the Dimension-aware term, and returns cited results.

This keeps a standard hybrid pipeline intact and adds one stage. Hypothetical Document Embeddings (HyDE; Gao, Ma, Lin, and Callan, 2023), which generate a hypothetical answer and retrieve by its embedding, are an escalation, not a default, and the one generation call the fast path can make: at most one per query, by a small model, when the top retrieval scores fall below a confidence threshold; Section 10.1 tests whether it earns its place. The read path’s decisions, the Router’s choice of path (Section 8) and the query’s Domain scope (Section 2), are calls to a decision model: a small-tier model class that takes a state and typed questions (a choice among listed options, a score on a described scale, or a yes/no), evaluates them together, and returns a probability or score per answer without writing text. Its output is confined to the declared options,

which prevents malformed answers, not wrong ones; calibration is tested in Section 10.6. The read path is therefore a fixed set of small calls: one query embedding, one decision-model call for the Router and Domain scope, the cross-encoder over a fixed candidate depth, at most $|A_D|$ projection calls per Domain in scope, and at most one small generation call. None scales with the size of the knowledge base, and none uses the reasoning model.

The slow path is reasoning and artifact generation. For reason it retrieves across the Domains in scope, derives or revises Claims, searches for evidence for their preconditions, projects them and runs the contradiction check and propagation of Section 3.4, returning the Claims with their justifications and standing; for author it then renders an audience-specific artifact from the Claims and their Signals (retrieval-augmented generation; Lewis et al., 2020) and validates citations. Its committed outputs are inputs to future searches; whether this compounds faster than staleness and repays the writes, Section 10.4 tests.

The slow path maintains the structure as a materialized view under deferred maintenance, which lets the view lag its base data between refreshes (Colby, Griffin, Libkin, Mumick, and Trickey, 1996): changes accumulate for Δ and are applied at commit. The consistency statement this entails is that the fast path reads the structure as of the last committed batch, and since a batch commits atomically (Section 3.4), never a Claim before its projection and check or a half-propagated standing change. Staleness is bounded in versions by one batch and in time by Δ plus the batch’s processing time, which Section 10.3 measures. Processing time itself is not bounded: a batch that outlasts Δ delays the next flush, writes queue, and staleness grows with the queue for as long as writes outpace processing; coefficients and active Dimension sets change only at cycle refits (Section 6). Reads within a session are monotonic, since commits only advance. Read-your-writes holds for the slow-path run that wrote a Claim, not for a fast-path read issued before the batch commits (Terry et al., 1994); that window is the staleness bound. Two things cross the boundary. From slow to fast: committed batches. From fast to slow: bookkeeping that schedules later writes, namely retrieval marks for re-derivation (Section 3.4), stale Claims queued for re-projection (Section 5.5), and the escalation log. The fast path writes no structure.

Why derivation needs the expensive model. Classification, contradiction labels and, after distillation, projection already run on small models. What remains is derivation: composing a proposition no single antecedent entails, from objects that may sit in different Domains, while searching for evidence for its preconditions. Its output is persisted, so an error in it is paid on every read that returns it, and it is the one write step with no cheap verifier. A Signal is checked against its span, a contradiction label by its precision bar, a projection by agreement between projectors, but a joint check answers a stricter question than derivation poses: derived Claims are defeasible judgements their antecedents support without entailing (the partnership Claim says “may”), so a joint-entailment check rejects them by definition or, relaxed to support, becomes the judgement it was meant to verify. A cheap deriver’s errors would pass unchecked into persisted structure, which is why derivation, and only derivation, gets the strongest model. Section 10.2 runs the alternative of Section 1 with the slow path’s reasoning model, so that comparison isolates persistence from model strength.

One item of evidence, followed through every layer, with supposed values:

- **Entry.** A partner profile arrives from the relationship records at time t_0 , authored by the partner manager.
- **Signal.** Extraction yields “Consortium Y operates within several Kenyan counties”, with its extractor, extraction time, span, and an `observed_at` taken from the profile’s date.
- **Domains.** The classifier scores it above threshold for Kenya and Organizational Relationships, and it is indexed in both.

- **Claim.** An author request about Foundation X’s county program runs the slow path, which derives the partnership Claim of Section 2 with one justification of the four Signals, this one among them. The precondition that the program is still open is searched for and found as a Signal, which joins the justification. H is the union of the antecedents’ Domains; standing is proposed.
- **Dimension.** In the same batch the Claim is projected on implementation feasibility; suppose the projector returns feasible with relevance 0.9, recorded under the Dimension’s current version. The check finds no conflict, the antecedents come from distinct Entries, and the automatic rule accepts the Claim. The batch commits.
- **Fast path.** A later request, which partners can implement in Kenyan counties, is labelled retrieve. Suppose the query projects to ready, the top level, with relevance 0.8: $\delta_d = 1/3$, so $\phi_d = 0.9 \cdot (1 - 1/3) = 0.6$ enters the score with weight $\beta_{D,d}$, and the Claim is returned with its citations and no generation call.
- **Change.** The same source later supersedes the Signal. Propagation retracts the partnership Claim, whose only justification has lost an antecedent, and every Claim justified only through it; the next read no longer returns it; and because it was retrieved in the last thirty days, re-derivation is scheduled, writing a superseding Claim if the remaining evidence supports a different route.

8 Router-Controlled Execution

A Router maps each request to one of retrieve, reason, or author, so users need not choose a path. It is a decision-model call (Section 7) whose state is the request text, the active workspace and artifact and the last few turns, and whose question is a choice among the three labels; the most probable label is taken unless its probability falls below a threshold set on validation data (Section 10.6), when the next slower path is taken, on the judgement that a shallow answer costs a user more than a slow one, which the two misroute rates of Section 10.6 let a deployment check.

The Router decides what computation occurs, the Domain classifier which context an object belongs to, and the Dimensions along which knowledge is compared; keeping them separate lets each be evaluated on its own (Sections 10.1, 10.5 and 10.6).

9 Related Work

Conceptual spaces. Gärdenfors (2004) supplies the representational level the Dimensions occupy (Section 4). Bechberger and Kühnberger (2017) formalize the framework, defining betweenness from the metric; because convex concepts combined across domains by the city-block metric cannot represent correlations between domains, they replace Gärdenfors’ convex regions with fuzzy star-shaped sets; Section 5.1 states which of these commitments the contract keeps.

Retrieval granularity and precomputed structure. Chen et al. (2024) find that proposition-level retrieval units outperform passages for retrieval and downstream question answering at a fixed budget, which supports Signals and also cuts the other way: a Claim layer above propositions may add little, which is why the Signal-only baseline of Section 10.2 is a proposition-level retriever. RAPTOR (Sarthi et al., 2024) recursively clusters and summarizes chunks into a tree at index time, another write-expensive, read-cheap design, but builds it once rather than maintaining it as

evidence arrives; with the Entry corpus frozen during the replay of Section 10.4 it would be flat there, so it is not an arm.

Graph-based retrieval-augmented generation and temporal graphs. GraphRAG (Edge et al., 2024) is the nearest existing system to this design’s write side among batch indexers. Its indexer uses an LLM to extract entities, relationships and, with a secondary prompt, claims about those entities (subject, object, type, description, source span, start and end dates), then pregenerates community summaries that answer global questions. Those claims are the nearer analogue to Claims, but are produced once from a text unit, with no justification beyond their source, a status fixed at extraction rather than a maintained standing, and no revision when the source changes. LightRAG (Guo, Xia, Yu, Ao, and Huang, 2025) updates its graph incrementally but likewise keeps no standing or justification sets. GraphRAG’s pipeline is a baseline for the Claims layer in Section 10.2, and an entity graph built as its indexer builds one underlies the graph-augmented arm of Section 10.1.

Zep (Rasmussen, Paliychuk, Beauvais, Ryan, and Chalef, 2025) is the nearest deployed system on the property GraphRAG lacks. Its Graphiti engine, a temporal knowledge graph for agent memory, ingests episodes incrementally and non-lossily, extracts relational facts as edges stamped with when the fact held and when the system believed it, and, on temporally overlapping contradictions, invalidates the affected edges as of the invalidating edge’s valid-from time. That is incremental supersession-by-contradiction, closer to Sections 3.3 and 3.4 than any other cited system; the system differs in three ways: Claims carry justification sets that may include other Claims, so new evidence acts by propagation rather than pairwise edge invalidation; a contradiction produces a contested state rather than a win for the newer fact, ended by a person or by a change in support, supersession being an explicit edge; and Dimensions give derived statements a geometry a fact graph lacks. Section 10.4 uses a Graphiti-style arm for comparison.

Contradiction detection. De Marneffe, Rafferty, and Manning (2008) studied the classifier’s task, finding contradictions in text, and found it harder than entailment: their system reached about 23% precision and 19% recall on the RTE-3 test set, and they aimed at high precision because contradictions are rare in real text. Modern NLI models reach high accuracy on standard benchmarks (He, Liu, Gao, and Chen, 2021), but that shape, both rates low and performance uneven by type, is why Section 10.3 sets separate precision and recall bars, reports candidate recall separately, and compares the small classifier with a frontier labeller.

Truth maintenance. Doyle’s truth maintenance system (1979) records justifications for beliefs and revises their status when assumptions are contradicted; de Kleer’s assumption-based TMS (1986) labels nodes with the assumption sets under which they hold, so inconsistent contexts can be explored in parallel. The standing model of Section 3.4 is a monotone, well-founded subset of Doyle’s network, and de Kleer’s labels, keyed by the unevidenced preconditions in context, are the extension the design does not adopt. The nearest database analogue is incremental maintenance of recursive views, by counting derivations or by delete-and-rederive (Gupta, Mumick, and Subrahmanian, 1993); acyclic justifications let one topological pass replace the rederive step.

Belief revision. The AGM postulates (Alchourrón, Gärdenfors, and Makinson, 1985) characterize rational contraction and revision of logically closed theories. The Claim set is not closed under consequence and the system implements no AGM operator, but AGM is relevant as a standard: the retraction behavior of Section 3.4 should remove exactly the Claims whose every justification depends on withdrawn evidence and nothing else, which is the dependency-network analogue of minimal change, and Section 10.3 checks that property directly.

Argumentation. Dung (1995) defines argumentation frameworks as arguments with an attack relation, with extension semantics that determine which arguments are collectively acceptable. The `conflicts_with` edge is an attack relation, and the contested state is what results from declining to compute an extension and surfacing the conflict instead. If chains of conflict become common,

grounded semantics is the candidate rule for computing standing automatically, as future work.

Provenance. PROV-DM (Moreau and Missier, 2013) describes provenance with entities, activities and agents. The records of Sections 2 and 3.1 map onto it: Entries, Signals and Claims are entities; extraction and derivation are activities associated with an agent, the extractor or deriver identifier or a person; a justification is a set of derivations from its antecedents, and a Domain’s `derived_from` link is a derivation; a `supersedes` edge is a revision; `withdrawal` is an invalidation. Micropublications (Clark, Ciccarese, and Goble, 2014) model a claim over evidence, from a statement with its attribution up to a statement with its supporting argument, evidence and challenges; a Claim with its justifications and `conflicts_with` edges is the maximal form, and what the system adds is maintenance, a standing recomputed when support changes.

Ontologies and knowledge graphs. The system is not an ontology: Domains have no class hierarchy, Dimensions carry no entailment, and the graph edges of Section 4 have no reasoning regime. Ontology evolution is nonetheless the reference for Dimension versioning: Noy and Klein (2004) argue that it differs from schema evolution and that compatibility between versions has several dimensions, which is why Section 5.5 separates changes that invalidate stored positions from changes that alter only comparisons, and records the version under which each position was made.

Organizational knowledge management. Nonaka (1994) describes organizational knowledge creation as a continuous dialogue between tacit and explicit knowledge; this design touches only the explicit side. Alavi and Leidner (2001) frame knowledge management systems as support for the creation, transfer and application of organizational knowledge; their repositories hold documents people codify and keep current, with no recorded support or standing. The Claims here codify interpretations that would otherwise live in people’s heads or one-off documents, but a model writes them and the network keeps their standing, leaving people to accept Claims and decide conflicts. Agent memory is the nearest automated analogue: generative agents (Park, O’Brien, Cai, Morris, Liang, and Bernstein, 2023) synthesize observations into reflections stored with pointers to the memories they cite, a kept interpretation with a justification, but do not revise them when cited memories are contradicted, the property Sections 3.3 and 3.4 add; Zep revises facts but keeps no justification sets over them. Section 10.4 charges the human cost of codification, expected to matter because Paulheim (2018) puts a manually curated statement at 2 to 6 US dollars, far above an automatic one.

Hybrid retrieval and citation evaluation. The fast path’s base, lexical and dense retrieval fused and reranked by a cross-encoder, is the base of every retrieval arm in Section 10. ALCE (Gao, Yen, Yu, and Chen, 2023) defines citation recall (whether each statement is fully supported by the passages it cites) and citation precision (whether each cited passage supports its statement), computes both with an NLI model, and reports strong correlation with human judgment; Section 10.2 adopts these metrics by name.

Factor graphs and sheaves (intuition only). Factor graphs (Kschischang, Frey, and Loeliger, 2001) compute marginals by local message passing but need a probabilistic semantics a discrete standing lacks, and DeepDive (Zhang et al., 2017) learned their weights from labels the system has no source for; the dependency network supplies the locality directly. Phillips (2018) describes generalization as patching local knowledge into a coherent whole, where sections agree on overlaps or fail to glue; that picture motivated recording disagreement between Domains, and its only operational content is the `conflicts_with` edge.

What is new, and what is assembly. Almost every part is known, from justification-based truth maintenance (Doyle, 1979) and interpretable quality dimensions (Gärdenfors, 2000) to hybrid retrieval, learning to rank with an L_1 penalty, NLI-based contradiction detection and incremental invalidation of extracted facts. The paper claims as new only their composition, at three points: truth maintenance over model-derived propositions whose justifications mix evidence with other

derived propositions, so that model-written interpretations carry a maintained standing; validated, versioned interpretable axes that serve both the reranker and the contradiction check’s candidate filter; and the write/read thesis stated with a ledger that can refute it. The rest is assembly, and Section 10 tests the assembly against its parts.

Counter-arguments. The strongest alternative design is the one Section 1 names: no persisted layer, strong hybrid or graph-augmented retrieval, and per-request reasoning by a capable model. Three arguments favour it: falling model prices shrink the read-side saving; longer contexts let per-request reasoning see more evidence; and maintained structure needs human attention organizations may not supply. The ledger states how break-even moves with prices (Section 10.4), a larger context neither keeps nor checks a result, and the human cost lines answer the third. Caching per-request answers is the cheapest competitor to persistence, but a cache is keyed by request, so it does not carry an interpretation to differently phrased requests or revise it when evidence changes.

10 Evaluation Plan

None of the experiments below has been run. Each subsection names the bet, baseline, metric, test set construction, success threshold and drop condition. The success bars, failure bounds, sample sizes, stopping rules and drop rules were registered publicly before any data existed, as version 3 of a registration whose amendments are public, do not change once held-out data has been scored, and any deviation is reported against that registration (Miracle, 2026). Values the text calls initial settings (k , Δ , K , the τ thresholds, and the thresholds of Section 5.4) are system parameters, not success bars: they are tuned on validation splits only and frozen before the first held-out look. Drop rules are binding: a component that meets its drop condition is removed in the form stated.

Shared protocol. Experiments run on at least three pilot workspaces with distinct organizational subject matter. Query sets combine anonymized queries from usage logs with queries written by domain annotators; each workspace contributes at least 150 general queries, meaning queries not written for one component. Relevance judgments are graded (0 to 3) by two annotators per query over pooled top-20 results from every system under comparison (Zobel, 1998); inter-annotator agreement is reported as Cohen’s κ (Cohen, 1960) and queries with $\kappa < 0.4$ are re-adjudicated. Each query set is split 40/60 into a validation split (for fitting $\beta_{D,d}$ and μ , running the Useful screen, and setting thresholds) and a held-out test split that nothing is tuned on. Per-Domain fits require at least 30 validation queries scoped to the Domain (Section 6); Domains below that count are reported as unfit and excluded from the Dimension-aware arms for that workspace; workspaces are chosen so that at least their three largest Domains meet the minimum, and where the logs fall short annotators write validation queries for those Domains until each does. Held-out queries scoped only to unfit Domains still count in the pooled 10.1 primary, which they dilute, and the per-Domain breakdown reports it. Statistical significance uses a paired bootstrap over queries (Smucker, Allan, and Carterette, 2007), one-sided, unless a test names another procedure. Queries are deduplicated on normalized text before splitting; queries both annotators judge unanswerable from the workspace, and queries the organization withholds for personal data, are removed and counted; splits are by user session, so one session never spans two splits.

Annotation protocol. Annotators are staff of the pilot organization who know the subject matter, blind to arm, working from written instructions whose operative definitions are these. Relevance: 3, the item answers the query or is required to answer it; 2, it answers part of it; 1, it is on the query’s topic but does not help answer it; 0, otherwise. A contradicting Signal is one sentence that, if true, makes the target Claim false under the target’s own context, shares its named entities, and copies none of its other wording; a distractor is a sentence on the same entities consistent with the target.

Domain labels are every Domain whose description the Signal falls under. Router labels are retrieve when stored items answer the request, reason when it asks for analysis or judgement without a document, and author when it asks for a document. Artifact requests written by annotators name an audience and a deliverable. Preference raters judge coverage (every part of the request is addressed), correctness (no statement is contradicted by the workspace’s evidence), and coherence (statements agree with each other and are ordered for the audience), then state one overall preference. Quality control: before production each annotator labels 20 calibration items with adjudicated answers and must reach $\kappa = 0.6$ against them; 10% of production items are hidden calibration items, and a batch whose annotator falls below 80% agreement on them is relabelled; every item is labelled twice, agreement is reported as Cohen’s κ per task, and disagreements on items with $\kappa < 0.4$ go to a third annotator.

Leakage. Three routes are closed. Claims derived for a request in a held-out session would put the test into the index, so Claims are built for evaluation only from requests in other sessions, and any Claim whose derivation record names a held-out session is removed before scoring. The one exemption is the replay of Section 10.4, which measures that reuse: its reads are retrieve-labelled requests, which write nothing, so no read’s own derivation enters the index, and the coefficient fit sees the replayed Claims’ text but no held-out judgment. Synthetic queries from held-out Claims leak the Claim layer’s vocabulary, so they may supplement but never replace judged queries, with results reported with and without them. Tuned components use only validation material, and no model judges an item it generated (the ALCE judge differs from the generator). Queries answerable only from Entries public on the web, and so possibly in pretraining data, are reported separately (exploratory).

Matched budgets. Every arm with tunable settings gets the same tuning budget on the validation split, 20 configurations sampled from its documented settings, and the same candidate depth, so no baseline runs at defaults against a tuned system (Dodge et al., 2019). Arms that call a reasoning or extraction model use the slow path’s model tier. Quality is compared at matched tuning budgets, not matched spend: the system’s write spend, which no baseline incurs, is charged in the ledger of Section 10.4, where cost is compared at parity of quality under the cost-accounting rule below.

Success rules, power, and multiplicity. Every comparative test below states a design effect, the true difference the test is powered to detect, and sizes its sample so that a difference of that size is detected with 80% power. The 80% floor is the IR convention (Sakai, 2016) rather than the 90 to 95% of registered reports: a missed effect keeps a component out rather than harming anyone, and 90% would enlarge every judged sample by about 39%. The per-query standard deviation of 0.15 for nDCG@10 differences budgets annotation and is replaced by the internal-pilot estimate below. The success rule is a positive, significant observed difference; the design effect is not a second bar, since requiring both would roughly halve power. The bar is not zero in practice: at the nominal one-sided 0.03 of each look, significance requires an observed difference of at least $z_{0.97}/(z_{0.95} + z_{0.80}) \approx 0.76$ of the design effect at the planned size, and about 0.53 of it at the cap of twice that size. A smaller observed gain can pass, so every gain is reported with its confidence interval. nDCG@10 is averaged because the decision concerns the expected per-query gain; values are bounded, the bootstrap assumes no normality, and the sign of per-query differences is reported as an ordinal check (exploratory). Each bet has one primary clause tested at one-sided $\alpha = 0.05$ or as the bet states; its other significance clauses are secondary and corrected as one family by Holm’s step-down procedure (1979). A primary result that is positive but not significant triggers one prespecified extension of the sample by the same size, and the pooled sample is tested once more; as a two-look procedure, each look is tested at a nominal one-sided 0.03 (Pocock, 1977), holding the overall type I error near 0.05. Samples are sized at $\alpha = 0.05$, so power at the first look alone is about 73%, and the 80% floor is met over the two looks. Secondary clauses are tested once, on the

sample at which the primary is decided. A primary that is still not significant after extension has failed, and a result at or below zero fails without extension.

Absolute bars and labels. A bar on a single proportion names a failure bound p_0 and a design value p_1 ; it is passed when an exact one-sided binomial test rejects $p \leq p_0$ at $\alpha = 0.05$, with the sample sized for 80% power at p_1 , one look, and no extension. The primaries of Sections 10.2 and 10.4 are tested once, at one-sided 0.03, with no extension. Ties never count as success: every clause needs a strictly positive significant difference or a rejected bound, except the non-inferiority clauses of Sections 10.1 and 10.5 and the fidelity guard of Section 10.2, whose margins are stated and which pass only when a confidence bound clears the margin. Primary and secondary clauses are confirmatory; per-workspace and per-Domain breakdowns, trends, latency percentiles and side-by-side generator comparisons are exploratory.

Variance re-estimation. For each primary comparison that cites this rule, the standard deviation of per-query differences is re-estimated once before the first look, from an internal pilot of the comparison’s first 50 judged held-out queries pooled across workspaces (Wittes and Brittain, 1990); the stage size and the extension are recomputed from it. The interim step reports only the standard deviation, never the mean difference or any test. The recomputed size is never below the planned size, a restriction under which the type I error inflation of the naive t-test is often negligible (Wittes, Schabenberger, Zucker, Brittain, and Proschan, 1999), and never above twice it. Unlike adding topics until power is reached, which biases results toward significance (Webber, Moffat, and Zobel, 2008), this is one prespecified re-estimation, floored, capped and blind to the mean. Shortfall rule: when the cap, or the number of judged queries available, stops the pooled held-out count short of what the re-estimated standard deviation requires, the design effect is raised to what the available count can detect at the same power, and the raised figure is reported before scoring.

Cost accounting. Every model call in every arm is logged with model and tokens, and costs are reported as tokens per model tier and at the day’s list prices. A request’s cost, in any arm, is every model call it triggers, including the writes it causes. Costs are compared only between arms at quality parity or better; a cheaper, worse arm is not a cost win.

10.1 Dimension-aware reranking

Bet: the Dimension term in Section 6 improves ranking over a strong retriever that has no access to conceptual positions, and the improvement is attributable to the geometry rather than to the standing penalty that ships with it.

Arms, all sharing the same retrieval stage over Signals and Claims (BM25 and dense retrieval fused by reciprocal rank fusion, reranked by a cross-encoder; fusion as in Cormack, Clarke, and Büttcher, 2009): (i) the hybrid baseline as stated; (ii) hybrid plus the standing penalty μ fit alone on the validation split, with no Dimension features; (iii) the full reranker, hybrid plus μ plus the Dimension term with $\beta_{D,d}$ and μ fit jointly; (iv) a permutation control, identical to (iii) except that each Claim’s projection records are shuffled across Claims within the Domain before the coefficients are refit; (v) a Signals-only arm, the hybrid baseline with Claims removed from the index, the retrieval half of the alternative of Section 1; (vi) a global-retrieval arm, identical to (i) but unscoped by Domain; and (vii) a graph-augmented arm, entity-matched local retrieval over an entity graph built as in Edge et al. (2024) from the workspace’s Entries with the slow path’s model tier, whose entities, relationships and source Signals join (i)’s candidates before the same cross-encoder.

Primary metric: nDCG@10 (Järvelin and Kekäläinen, 2002) on the held-out split; secondary: Recall@50; MRR (Voorhees, 2000) is reported as exploratory only, since reciprocal rank is not interval-scaled and is not averaged into any decision. The Dimension gain is (iii) minus (ii); the standing gain is (ii) minus (i); the Claims-in-index gain is (i) minus (v); the system gain, full system

over the alternative’s retrieval, is (iii) minus (v); the Domain-scoping gain is (i) minus (vi); the graph gain is (iii) minus (vii). Primary clause: the Dimension gain, design effect 0.03 absolute nDCG@10, for which the pooled held-out general queries (about 270 across three workspaces) give at least 80% power at a planning standard deviation of 0.15, re-estimated under the variance re-estimation rule (Section 10). Secondary clauses, five in the Holm family: the system gain is positive and significant, powered for 0.03 (about 250 queries at the strictest Holm level, 0.01, within the pooled 270); in every workspace, the lower confidence bound on the Dimension gain at its Holm level lies above -0.03 , a non-inferiority margin equal to the design effect because a workspace’s sample cannot resolve a smaller one (with about 90 held-out queries per workspace, all three pass with about 80% probability when the true gain is 0.03); the Domain-scoping gain is positive and significant, powered for 0.05; the graph gain is positive and significant, powered for 0.03, which the pooled queries cover at the strictest Holm level; and the standing gain, positive and significant, powered for 0.03 on the same terms. The permutation control is the attribution test: shuffling removes all geometric information while μ is refit and survives, so the control is compared with (ii), and any gain of (iv) over (ii) is fitting noise; success requires that (iv) minus (ii) be at most half of (iii) minus (ii). Positive control, a check rather than an arm: on the validation split, an oracle feature equal to each Claim’s judged relevance, fit as a Dimension term, must beat (ii) significantly, or the fitting and scoring pipeline is broken and is repaired before any held-out look. Cost: the query projection calls must add no more than 10% to median fast-path cost per query for queries scoped to at most three Domains, which bounds the calls at $3K$; the bound is reported per scope size.

Drop conditions are separate for the two terms. The Dimension term is dropped if the Dimension gain fails under the shared success rule, or the permutation control retains more than half of it, or the cost bound is exceeded and the gain vanishes when query projection is disabled; dropping it also parks all Dimensions that were active only by the Useful screen. The standing penalty is kept if the standing gain is positive and significant and dropped otherwise, independently of the Dimension term. If the Domain-scoping gain is not positive and significant, scoping is removed from the fast path, and Domains remain only as the homes of Claims and Dimensions, with $w_q(d)$ taken over c ’s home Domains. If the system gain is not positive and significant, the quality half of the thesis fails on retrieval (Section 10.7). If the graph gain is not, graph-augmented retrieval matches the system and is reported as such. HyDE escalation is tested outside the Holm family, on the held-out queries where it fires: (iii) with escalation against (iii) without, by the same paired bootstrap at one-sided 0.05, design effect 0.05 nDCG@10, for which about 56 firing queries give 80% power. Unless at least that many fire and the difference is positive and significant, escalation is removed and the fast path makes no generation call.

10.2 The Claims layer

Bet: artifacts rendered from persisted Claims are better than artifacts produced without a persisted knowledge model, at comparable citation fidelity, and the improvement comes from the persisted model rather than from reasoning before rendering.

Baselines: (a) Signal-only generation, where the same retriever supplies proposition-level Signals directly to the same generator in one pass, which is the strongest form of the granularity argument from Chen et al. (2024); (b) a GraphRAG-style pipeline, where an entity graph with extracted entity claims, a step Edge et al. (2024) describe but did not evaluate, and community summaries is built over the workspace’s Entries and supplied to the generator; and (c) per-request reasoning, the slow path of Section 7 run with persistence disabled: the same retriever, the same reasoning model, and the same renderer derive interpretations for the request, use them, and discard them, writing no Claims, no projections, and no standing. Baseline (c) is the artifact half of the alternative of

Section 1, and it separates the persisted model from reasoning-before-rendering. An exploratory arm (d) runs the full system with derivation on the small tier, which checks Section 7’s argument that derivation needs the strongest model. Metrics: blinded pairwise preference on coverage, correctness, and coherence by two raters per pair, a third adjudicating disagreements, with pairs the third calls a tie excluded and reported; ALCE citation recall and precision (Gao, Yen, Yu, and Chen, 2023), computed with an NLI model and checked against human judgment on a 20% subsample; a precondition check, the fraction of cited Claims with non-empty context whose preconditions appear in the artifact; and cost per artifact request for every arm, counted as the shared protocol counts it.

Test set: 85 artifact requests per pilot workspace, sampled uniformly from author-labelled requests in the logs outside the replay window of Section 10.4 and topped up by annotator-written requests where a workspace has too few (the split is reported), spanning at least 10 Domains per workspace, with audience specified (engineering or stakeholder), so that at least 225 decided pairs per baseline comparison remain after tie exclusion; if more than 30 are ties, further requests are drawn in a random order fixed in advance until 225 pairs are decided, before any preference is tested, and the number drawn is reported. The size follows from the strictest level at which any preference clause is tested, 0.05/3 for the first Holm step, which also covers the primary’s single test at 0.03: a true 60% preference rate against a 50% null is then detected with 80% power by an exact binomial test on about 225 pairs, which at that size means observing 129 or more wins. As a positive control, the generator with no evidence at all must lose to (a) significantly; if it does not, the raters are not discriminating, and the comparison is rerun with recalibrated raters before anything is scored.

Primary clause: preference against baseline (c), by exact binomial test, with 60% as the design effect. Secondary: preference against (a) and (b). Guard, outside the Holm family: ALCE citation precision and recall non-inferior to the best baseline within 0.05, by one-sided 95% bootstrap bound over statements; a preference win that fails the guard is not a win. The precondition check must be 100% (it is a renderer requirement, and any miss is a defect). Cost per artifact is not a success condition here; the persistence overhead is judged in Section 10.4’s ledger. Decision rule: if the system is preferred to (c) significantly, and passes the guard, the persisted model contributes to artifact quality; otherwise, including when it beats (a) but not (c), the Claim layer adds nothing to artifacts that per-request reasoning lacks, the quality half fails on artifacts (Section 10.7), and artifact generation is dropped back to baseline (c), with Claims kept only if Section 10.1 keeps them for retrieval. Drop condition: if the system is not significantly preferred to (a), granularity did the work, which is the Chen et al. outcome, and the Claim layer is dropped in favor of investment in Signal extraction; when the system beats (c) but not (a), this drop condition governs, whatever Section 10.1 finds. Nothing is tuned here: arms use the settings tuned on the 10.1 validation split, and all requests are held out.

10.3 Standing and dependency tracking

Bet: the contradiction check and propagation of Section 3.4 detect conflicts and retractions that matter, cheaply, without regenerating Claims, and each branch of the check can be credited separately; and the entailment check of Section 3.1 decides the Signal/Claim boundary correctly.

Test set, two arms, both injected through the ordinary write path so that the runtime, not the experimenter, performs classification, projection, and the check, and both scored on `conflicts_with` edges by the branch tag that Section 3.4 records on every edge. In the Signal arm, for each pilot workspace, 100 accepted Claims are selected as targets; for each, an annotator writes one Signal that contradicts it and one distractor Signal on the same topic that does not, and the 200 Signals are injected in random order over a simulated week. This exercises the Signal branch of the candidate step (similarity alone, with the floor τ_s), the lifting of contradicting Signals into Claims, and the

classifier. In the Claim arm, targets are 100 accepted Claims each with at least one active Dimension on which $r_d \geq \tau_r$, so the position filter can apply; for each, the annotator writes a contradicting Signal and a Claim justified by it alone that opposes the target on such a Dimension, plus a distractor Signal and Claim, each pair written as a unit, and the runtime projects each Claim in its batch before the check. The justifying Signal will usually also be caught by the Signal branch; that is production behavior and is not suppressed, so the Claim arm is scored only on edges tagged as written by the Claim branch. Pooled over three workspaces each arm has 300 targets.

Baselines, on copies of each workspace receiving the same injections: the exhaustive check, which runs the same classifier against every accepted Claim in the Domain with no candidate step and no floor, as a global recomputation would; the same candidate pairs labelled by a frontier model instead of the small classifier; and, for the Claim branch, similarity-only candidates at the same k . The exhaustive check is also the positive control: if its own recall is below 0.7, the classifier, not the candidate step, is failing, and the result is attributed to it. The Signal arm is also rerun on a copy at $k = 50$ with no floor, for the drop condition below. For propagation, a global relabelling recomputes every label from scratch after each batch; the network’s labels must equal it exactly, with the consequence of the minimal-change check, and nodes visited per change are reported against the global count. The parameters k , τ_s , τ_r and τ_δ are set on 30 further targets per workspace per arm, injected the same way and disjoint from the scored ones.

Metrics, Signal arm: recall, the fraction of targets that acquire a Signal-branch edge to the lifted Claim of their contradicting Signal; precision, the fraction of Signal-branch edges written during the arm that join a lifted Claim to a Claim the annotator marked as contradicted by it; candidate recall, the fraction of injected contradictions whose target was among the k candidates after the floor τ_s ; contradictions removed by the floor; and model calls per injected Signal. Claim arm: the same on Claim-branch edges; the fraction of checks in which the position filter applied; candidate precision with the filter versus similarity alone at the same k , over checks in which the filter applied; and model calls per injected Claim, including projection. Both arms: wall-clock from write to the contested transition. Retraction, both arms: a minimal-change check in which, after withdrawing the Entries behind 20 accepted Claims, the set of Claims that become retracted must equal the set whose every justification depended on those Entries, with no extra retractions, every Claim contested only by one of them must leave contested, and restoring the Entries must return exactly those Claims to proposed, which Section 3.3’s rule that re-entered Claims are never auto-accepted guarantees in every workspace configuration. Automatic acceptance: annotators, blind to sample, judge whether the evidence supports 230 Claims the rule accepted in the replay of Section 10.4 (extended as that section states if it holds fewer) and, as baseline, 230 drawn at random from all Claims the slow path wrote, which is what accepting everything would accept; the samples overlap, which makes the comparison conservative. Entailment check: per workspace, 150 extracted Signals (is the proposition entailed by its span?) and 150 derivations (is the proposition entailed by no single antecedent?), drawn from ingest and the replay, labelled under the shared annotation QC and split 40/60 by Entry; the validation part sets any threshold, and each condition is scored once on the held-out part (about 270 pooled) by accuracy against the adjudicated label, with the false-pass rate reported. Baselines: the majority label and the frontier labeller on the same items. Contradictions occurring naturally in the logs, found by annotators in a sample of 100 Claims per workspace, are scored for recall as an exploratory check on the injected sets’ realism.

Primary clause: Signal-branch recall. Success, as absolute bars (Section 10): recall above $p_0 = 0.7$ with design value 0.8 (119 targets give 80% power) and precision above 0.6 with design value 0.7 (143 edges) on each branch’s own edges; candidate recall above 0.8 with design value 0.9 (82 targets) on each branch, with the floor removing none of the injected contradictions; the position filter raises candidate precision over similarity alone on the checks where it applied without lowering

candidate recall below 0.9; model calls per injected object at most $k + 1$ for the check, plus at most K projection calls in the Claim arm; median wall-clock to the contested transition under Δ plus one minute; minimal-change check exact in both directions; and, because a supports verdict adds a justification that decides survival under retraction, the precision of supports verdicts on a sample of 150 such justifications labelled by annotators above 0.7 with design value 0.8 (119 give 80% power), and the same bar for the 230 automatically accepted Claims, whose precision must also exceed the baseline sample’s by a one-sided two-proportion test at 0.05, design effect 0.10 (230 per sample give 80% power at 0.8 against 0.7); and entailment-check accuracy above 0.8 with design value 0.9 on each condition (83 items give 80% power), and above the majority label by McNemar’s exact test.

Comparisons with the baselines are secondary, by McNemar’s exact test over targets: if the exhaustive check’s recall exceeds the system’s significantly, the candidate step is losing contradictions the classifier would catch; if the frontier labeller’s precision exceeds the small classifier’s significantly, the classifier step moves to the frontier model, at the cost the ledger then carries.

Redesign conditions: Signal-branch candidate recall below 0.7 means k must rise, at linear cost, or τ_s must fall; if the position filter lowers Claim-branch candidate recall below 0.9, the filter is removed and that branch falls back to similarity alone; precision below 0.5 on either branch moves the classifier step to a stronger model or a human queue; a minimal-change check or global-relabelling comparison that is not exact is a propagation defect, fixed and rerun before any other clause here is scored. If automatic acceptance fails its bar or does not beat its baseline, the rule is disabled and every acceptance is made by a person, the ledger’s second computation in Section 10.4. If supports-verdict precision fails its bar, supports verdicts stop writing justifications and are logged for review only. If the entailment check fails either clause on a condition, the frontier labeller takes that condition over if it passes the same bar on the held-out part, and is used anyway if it is significantly more accurate by McNemar’s test; if both fail, the check is removed for that condition, Signals entering unchecked and flagged as such, or derivations losing the restatement check and every derived Claim staying proposed until a person accepts it. Drop condition: if Signal-branch recall at $k = 50$ with the floor removed is not significantly above 0.6 (design value 0.7; 143 targets give 80% power, and the arm has 300), automatic standing changes are disabled and standing becomes a purely human-maintained field.

10.4 Fast/slow compounding, staleness, and the cost ledger

Bet: accumulated slow-path work makes the fast path better and cheaper over time, the total spend on writes, maintenance, human curation, ingest checks, and reads falls below the alternative’s spend at realistic read/write ratios, and the structure does not go stale faster than it is maintained.

Two replay series are run on each pilot workspace, both starting from a fresh copy containing only its Signals. The m slow-path runs and the N reads come from one window of the workspace’s request log, in logged order and from held-out sessions only, so no replay read is in the validation split on which $\beta_{D,d}$ and μ are fit: requests labeled author or reason in Section 10.6 become the slow-path runs, and requests labeled retrieve become the reads, so the overlap between what is written and what is read is the workspace’s own and routing does not vary between arms. N is at least 72 per workspace, 216 pooled, which gives 80% power for the design effect below at one-sided 0.03 and a per-query standard deviation of 0.15; because the log window fixes the read set, the primary here is tested once, at 0.03, with no extension. The read set is judged only after the replay, so no internal pilot can be drawn from it; its size is recomputed before the replay from the internal-pilot estimate of Section 10.1, on the same workspaces and metric, and the realized standard deviation is reported with the result, the detectable effect being raised under the shortfall rule (Section 10) if it exceeds the value used. In the clean series only the slow-path runs change between checkpoints: the

N queries are replayed at $m = 0$ and after $m = 5, 10$ and 20 , with the Signal corpus, query set and Dimension set frozen. Coefficients $\beta_{D,d}$ and μ are fit once, on the validation split against the Claims present at the final checkpoint, and applied at every checkpoint; at $m = 0$ there are no Claims, so the Dimension term is inert there. Claims that first enter any arm’s top 20 at a checkpoint did not exist when judgments were pooled, so a post-replay pooling and judging pass is run before scoring. This series measures compounding and read cost. Positive control, a deliberate leak used only as a check: on a copy at $m = 20$, the reads’ own requests are run once as reason, and the fast path must then beat $m = 0$ significantly, or the replay or indexing pipeline is broken and is repaired before the clean series is scored. The injected series performs the same runs and, at each checkpoint, injects a contradiction set against the accepted Claims then present; it measures staleness and per-Signal ingest cost, kept apart so injected noise cannot pass for compounding.

Contradiction set construction. At checkpoint m , annotators select up to 100 accepted Claims from those that exist, and for each write one contradicting Signal, following the Section 10.3 Signal-arm procedure. With fewer than 20 accepted Claims at a checkpoint, staleness there is reported as not measurable rather than as zero.

Quality and staleness metrics at each m on the clean series: fast-path nDCG@10; HyDE escalation rate; and p50 and p95 query latency. On the injected series: the staleness rate, defined as the fraction of top-10 results that are Claims whose standing is accepted or proposed while an injected Signal contradicting them has been present for longer than the batch interval Δ . A contested Claim in the top 10 with its conflict surfaced is the design working and is not counted. Retracted or superseded Claims in any top 10 must number zero (Section 3.3); a non-zero count is a retrieval defect, not a staleness result. Primary clause: the gain in nDCG@10 from $m = 0$ to $m = 20$ on the clean series, positive and significant by paired bootstrap over the N queries, powered for a design effect of 0.03. Secondary: no checkpoint significantly below the checkpoint before it; the escalation rate falls; p95 latency does not rise. Because that rate is diluted when reads rarely reach the targeted Claims, the tested bar uses probes: after Δ has elapsed, each injected target’s own proposition is issued as a query, and probe staleness is the fraction of probes whose top 10 contains the target at accepted or proposed standing. Success on the injected series: probe staleness below 0.10, design value 0.05, by exact one-sided binomial test pooled at the final checkpoint (about 180 probes give 80% power; up to 300 are available), with the all-results rate reported at every checkpoint.

Arms for the injected series: the full system as specified, and a Graphiti-style arm (Rasmussen et al., 2025) ingesting the same Signals incrementally into a temporal fact graph that invalidates temporally overlapping contradicted facts; its probe staleness counts probes whose top 10 holds a contradicted fact not yet invalidated. Per-Signal ingest cost is measured for both: the system’s classifier and lifting calls, written c_{ing} , and the arm’s extraction and invalidation calls. If the full system’s probe staleness is not significantly below this arm’s, by a one-sided two-proportion test at 0.05 on the pooled final-checkpoint probes (about 110 per arm give 80% power at 0.05 against 0.15), at a per-Signal ingest cost no lower than the arm’s, the network is not earning its complexity on this axis, and, binding, the Signal branch of the check and lifting are removed, so Signals act on Claims only through supersession and withdrawal.

Read arms on the clean series. The fast path is compared with two read arms that hold no Claims: hybrid retrieval over Signals alone, arm (v) of Section 10.1; and the same retrieval followed by the reasoning model over the retrieved Signals, the most the alternative would spend on a read. Quality is compared on the same N judged queries at the reasoning arm’s evidence depth: with $|C_q|$ the Signals its answer cites, every arm’s list is truncated to its top $|C_q|$ and scored by nDCG and precision there. A baseline arm reaches parity when the system is not significantly better than it, deliberately absence of significance rather than an equivalence margin, the stricter rule against the system, since the cheapest arm the system cannot beat sets the read baseline. That arm’s per-query

cost is r_{base} ; if none reaches parity, it is the reasoning arm’s cost.

Cost ledger. Every arm serves the same workload: m slow-path requests, N reads, and the Signals ingested while a workspace produces m requests. The system is charged five lines. $W(m)$ is the cost of the m slow-path runs on the clean series. $M(m)$ is automated maintenance over the same period: re-derivation after relabelling, re-projection and version backfill, Stable re-checks, discovery and validation, and coefficient refits, logged on the injected series, prorated per evaluation cycle, and with its relabelling share rescaled from the injected contradiction rate to the rate of withdrawals and supersessions in the replay window. $m \sigma c_{\text{ing}}$ is ingest checking, where σ , Signals ingested per slow-path request, comes from the same logs as $\bar{N}$, and c_{ing} , the classifier and lifting cost per Signal, from the injected series; the alternative has no such line, since it only embeds and indexes Signals, as the system also does. $N \cdot r_{\text{fast}}$ is reads at the fast path’s mean per-query cost at $m = 20$, including query projection, the decision-model call of Section 7 at a per-call cost c_{dec} , and any HyDE escalation. $H(m) = L \sum_a n_a(m) t_a$ is human curation, over the actions the design requires of people: acceptance and re-acceptance after retraction, conflict resolution, review of Dimension replacements, pins and retirements, and Domain creation and revision. The counts n_a are the actions the replay leaves for people (Claims awaiting acceptance, recorded conflicts, queued Dimension reviews) plus the Domain creations and revisions in the log window, the initial authoring of the workspace’s Domains prorated over it, and the description revisions and reclassifications of Section 10.6; the minutes t_a are the median of 30 timed performances of each action by the pilot’s reviewers, and L is the loaded hourly cost of the reviewing role in the pilot organization, stated with the ledger. Domain creation, with its re-scoring of Signals, is also reported as its own line and is charged to the system although the alternative scopes by the same Domains, so that no curation cost is left out, and the ledger is computed twice, with automatic acceptance on and with every acceptance made by a person. It is also computed with each replayed request charged the cost of the path the Router, as deployed, sends it to, so that escalations and misroutes are paid for; the success condition must hold under both routings. For the alternative, each slow-path request is served by baseline (c) of Section 10.2 at its measured cost g_{base} and each read at r_{base} , so its artifact and read baselines are one system, with total $m \cdot g_{\text{base}} + N \cdot r_{\text{base}}$. The break-even read count is

$$N^* = \frac{W(m) + M(m) + H(m) + m \sigma c_{\text{ing}} - m \cdot g_{\text{base}}}{r_{\text{base}} - r_{\text{fast}}}$$

the reads over which the system’s extra write, maintenance, curation and ingest cost is repaid; N^*/m is the break-even read/write ratio. A uniform change in model prices moves N^*/m only through H , which does not scale with them, so as prices fall the human term dominates. The frontier-to-small price ratio moves W , g_{base} and, with the reasoning arm at parity, r_{base} , but not H , ingest or r_{fast} : with the reasoning arm at parity a higher ratio lowers N^*/m and a lower one can remove break-even, as the illustration shows; with plain retrieval at parity N^*/m rises with the ratio. N^*/m rises linearly in σ , with slope $c_{\text{ing}}/(r_{\text{base}} - r_{\text{fast}})$. The ledger reports N^*/m at the day’s list prices and on a grid of 0.25, 0.5, 1, 2 and 4 times the day’s frontier-to-small price ratio and 0.5, 1 and 2 times L , each against $\bar{N}$, and it reports the projection and ingest lines separately, which answers the question Section 5.5 leaves open. An illustration with supposed prices, not estimates (small call \$0.0005, frontier call \$0.025, reviewer \$60 an hour): suppose each slow-path request costs 20 frontier calls and 8 projections in W (\$0.504), \$0.05 in M , 30 seconds of review in H (\$0.50), and 30 ingested Signals at $k + 1 = 21$ classifier calls each (\$0.315), against g_{base} of 15 frontier calls (\$0.375); and suppose a fast-path read costs the equivalent of 30 small calls, mostly cross-encoder scoring with query projection within the 10% bound of Section 10.1, plus c_{dec} , against one frontier call and two small calls (\$0.026) for the reasoning arm. Then $N^*/m = 0.994/(0.011 - c_{\text{dec}})$: about 90 reads per slow-path request when c_{dec} is negligible, and no break-even once c_{dec} reaches \$0.011.

Halving the frontier price removes the break-even, since the reasoning read falls to \$0.0135, below the fast path’s \$0.015; doubling it lowers N^*/m to about 31, or 32 if M , being model spend, doubles too; doubling the labor rate raises it to about 136; and if plain retrieval reaches parity there is no break-even at any of these prices.

The observed ratio $\bar{N}$, read queries per slow-path request, is measured from pilot logs, not assumed. The success condition is $N^* \leq \bar{N} \cdot m$ at $m = 20$: the structure pays for itself within the reads a workspace actually performs. It is tested as a bound: the one-sided 95% bootstrap upper bound on N^* , resampling slow-path requests, reads and timed actions together, must lie at or below $\bar{N} \cdot m$, and a resample whose denominator is at or below zero counts as no break-even. “ N^* above $\bar{N} \cdot m$ ” below and in Section 10.7 means that this bound is above it. The test is sized before the replay by simulation from internal-pilot costs and the pilot-log $\bar{N}$, for 80% power at a minimum detectable margin of a true N^* at half of $\bar{N} \cdot m$: the reads and the slow-path requests costed are increased up to three times the planned counts, the extra reads and runs being costed only, never judged, so the quality samples and their cap are unchanged, and the runs drawn from the same log window also fill the acceptance samples of Section 10.3. Sizes are reported before scoring, and a shortfall at the cap moves the margin under the shortfall rule. The condition $r_{\text{fast}} < r_{\text{base}}$ is necessary for any break-even and is reported as a sanity check, not as a success.

Drop conditions: no significant gain by $m = 20$ on the clean series means slow-path outputs are not being reused; probe staleness failing its bar means the structure decays faster than propagation maintains it, the weaker failure of Section 1; and N^* above $\bar{N} \cdot m$ under either routing, or no break-even at all, means the cost half is false even if the quality half holds. Any of these returns the system to the alternative.

10.5 Dimension discovery and projection

Bet: the discovery pipeline of Section 5.4 produces Dimensions that rank as well as Dimensions people author and that people recognize as meaningful, and projection can be done by a small model.

Baselines: human-authored Dimensions, and for projection the frontier projector itself. Before any held-out query is seen, one domain annotator per Domain authors up to K Dimensions under the contract of Section 5.1 from the Domain description and a sample of its Claims, in a session of at most two hours timed for the ledger of Section 10.4; these pass the same validation and are projected by the same projector. Metrics and test set: the Dimension gain of Section 10.1 on its held-out queries, once with the discovered active sets (arm (iii)) and once with the human-authored ones (arm (iii-h), coefficients refit); meaningfulness, where two raters blind to each promoted Dimension’s source answer “would you use this axis to compare these Claims?” on ten projected Claim pairs, a Dimension counting as meaningful when both say yes on a majority, κ reported; the two candidate generators’ promotion and meaningfulness rates side by side (exploratory); and agreement, r_s , between a small projector fine-tuned on the frontier bootstrap and the frontier projector, on 30 to 200 held-out Claims per Dimension, with the ratio of per-projection costs.

Two outcome-neutral checks come first: projecting each Dimension’s own prototypes must return their assigned values, or the projection pipeline is broken; and the human-authored set must itself help, (iii-h) minus (ii) positive and significant, or the comparison has no assay sensitivity, and discovery stands or falls with the Dimension term in Section 10.1. Primary clause: discovered Dimensions are non-inferior to human-authored ones in ranking, that is, the one-sided 95% paired-bootstrap lower bound on (iii) minus (iii-h) lies above -0.03 nDCG@10, and, since that margin equals the whole 10.1 design effect, the discovered sets must also retain at least half the human-authored gain, (iii) minus (ii) at least half of (iii-h) minus (ii), a point comparison like the permutation

control's. The claim is parity with human authoring at lower human cost; a zero difference passes only if the bound clears the margin, and the pooled 10.1 queries give 80% power for a 0.03 margin at a true difference of zero. Secondary, as absolute bars: meaningful promoted Dimensions above 0.5 with design value 0.75, for which 23 Dimensions give 80% power, with discovery run for up to three cycles to reach 40 pooled and fewer than 23 after three cycles counted as a failure; and the share of active Dimensions on which the small projector meets the Measurable threshold, $r_s \geq 0.7$ against the frontier projector, above 0.6 with design value 0.85 (21 Dimensions), at a per-projection cost no more than one tenth of the frontier projector's. Drop conditions: if the primary or the meaningfulness bar fails, automatic discovery is removed and Dimensions are human-authored per Domain, with validation and projection unchanged; if the projector bar fails, projection stays on the larger model and c_π becomes the binding constraint on the number of active Dimensions, lowering K and raising $W(m)$ in the Section 10.4 ledger.

10.6 Router and Domain classifier

Bet: the two routing decisions the fast path depends on are accurate enough that the failures of Sections 10.1 through 10.4 can be attributed to the components under test rather than to misrouting.

Router. Test set: 300 held-out logged requests per workspace, 900 pooled, labeled retrieve, reason, or author by two annotators, κ reported; a further 200 per workspace, disjoint by session, are the validation part that sets the Router's confidence threshold and fits the embedding baseline, and the labels also route the replay of Section 10.4. Metrics: accuracy, and the two misroute rates separately, since a slower path wastes compute and a faster one gives a shallower answer. Baselines: the majority label, and the embedding baseline, a multinomial logistic regression over request embeddings fit on the validation part under the matched tuning budget; a frontier model prompted with the label definitions is reported as a reference (exploratory), since the Router is meant to be cheap, not to match it. Outcome-neutral checks: the embedding baseline must beat the majority label, or the labels are noise and the test is uninterpretable; and calibration is tested rather than assumed, by expected calibration error over ten probability bins, which must be below 0.05 for the confidence threshold to be used at all, the top label being taken otherwise. Primary clause, as an absolute bar: accuracy above 0.85 with design value 0.9 (about 280 requests give 80% power; the pooled set has 900). Secondary: accuracy above the embedding baseline by McNemar's exact test, design effect 0.05 (at a discordance rate of 0.15 about 370 requests give 80% power, and about 470 at the stricter Holm level); and the shallow-misroute rate below 0.10 with design value 0.05 (about 180 requests). Drop condition, applied before any downstream result is interpreted: if the primary or either secondary fails, the Router is removed; the embedding baseline replaces it if it passes the primary bar, and otherwise the user chooses retrieve, reason or author.

Domain classifier. Test set: 800 Signals per workspace, labelled by two annotators, κ reported per Domain, and split 40/60 by Entry like the query sets; thresholds θ_D , the per-Domain check below and any description revision use the validation part, and the clauses are scored once on the held-out part, stratified so each Domain has at least 40 Signals there. Baselines: the majority Domain; nearest-description similarity with one global threshold, which must itself beat the majority Domain as an outcome-neutral check; and one-vs-rest logistic regression over Signal embeddings fit on the validation part, the stronger of the two embedding baselines on that part being the comparison below; a prompted frontier model is reported as a reference (exploratory); calibration of the returned probabilities is tested as for the Router before θ_D is set from them. Metrics: per-Domain precision and recall, and macro-F1. Primary clause: macro-F1 above 0.80 by one-sided 95% bootstrap bound, resampling Entries so that Signals from one Entry move together, design value 0.85; with at least 40 held-out Signals in each of 30 or more pooled Domains, a binomial approximation at the design

value puts the standard error of macro-F1 near 0.01, so power is far above 80%. Secondary: macro-F1 above the stronger embedding baseline by the same paired bootstrap, design effect 0.04, which with independent errors has a standard error near 0.014 and about 85% power; the realized standard error is reported, and the detectable effect raised under the shortfall rule if it is larger. Per-Domain recall of at least 0.75 matters because a Signal the classifier misses is invisible to every query scoped to that Domain, but per-Domain samples are too small to test it, so it is an exploratory check on the validation part: a Domain below it has its description revised and its Signals reclassified before Section 10.1 runs. Drop condition: if the primary or secondary fails, the classifier is removed from the fast path, Domain scope stops being a hard filter, and the workspace runs as arm (vi) of Section 10.1.

10.7 What falsifies the thesis

The quality half of the thesis fails if either comparison with the alternative fails: if the full system’s gain over the Signals-only hybrid arm is not positive and significant in Section 10.1, or if it is not significantly preferred to per-request reasoning without persistence in Section 10.2, the persisted structure is not improving that part of what users see. The cost half fails if Section 10.4’s ledger, with maintenance, curation and ingest charged and the read baseline set to the cheapest arm at parity, finds N^* above $\bar{N} \cdot m$ under either routing, or no break-even at all. The thesis fails in the weaker sense if probe staleness fails its bar in Section 10.4. A component that meets its own drop condition is removed, and the thesis is judged on the rest; a framework that fails remains research, not architecture.

11 Conclusion

The organizing asymmetry is that expensive intelligence writes structure and cheap intelligence reads it, and most knowledge stays dormant until new evidence or a conflict makes it matter. The design commits to specific mechanisms so that it can be wrong in specific ways, and Section 10 gives each a baseline, a bar fixed in advance and a binding drop condition. Both halves of the thesis are tested against a system that persists nothing and reasons per request; if it does as well, or as well for less, the answer is to build it.

References

- Alavi, M., & Leidner, D. E. (2001). Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, 25(1), 107–136.
- Alchourrón, C. E., Gärdenfors, P., & Makinson, D. (1985). On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *The Journal of Symbolic Logic*, 50(2), 510–530.
- Alshaikh, R., Bouraoui, Z., & Schockaert, S. (2020). Hierarchical Linear Disentanglement of Data-Driven Conceptual Spaces. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)*, 3573–3579.
- Bechberger, L., & Kühnberger, K.-U. (2017). A Thorough Formalization of Conceptual Spaces. In *KI 2017: Advances in Artificial Intelligence, Lecture Notes in Computer Science 10505*, 58–71. Springer.

- Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A Large Annotated Corpus for Learning Natural Language Inference. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 632–642.
- Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to Rank Using Gradient Descent. *Proceedings of the 22nd International Conference on Machine Learning*, 89–96.
- Chatterjee, U., Gajbhiye, A., & Schockaert, S. (2023). Cabbage Sweeter than Cake? Analysing the Potential of Large Language Models for Learning Conceptual Spaces. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 11836–11842.
- Chen, T., Wang, H., Chen, S., Yu, W., Ma, K., Zhao, X., Zhang, H., & Yu, D. (2024). Dense X Retrieval: What Retrieval Granularity Should We Use? *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 15159–15177. (First posted as arXiv:2312.06648, 2023.)
- Clark, T., Ciccarese, P. N., & Goble, C. A. (2014). Micropublications: A Semantic Model for Claims, Evidence, Arguments and Annotations in Biomedical Communications. *Journal of Biomedical Semantics*, 5, 28.
- Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1), 37–46.
- Colby, L. S., Griffin, T., Libkin, L., Mumick, I. S., & Trickey, H. (1996). Algorithms for Deferred View Maintenance. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 469–480.
- Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 758–759.
- de Kleer, J. (1986). An Assumption-based TMS. *Artificial Intelligence*, 28(2), 127–162.
- de Marneffe, M.-C., Rafferty, A. N., & Manning, C. D. (2008). Finding Contradictions in Text. *Proceedings of ACL-08: HLT*, 1039–1047.
- Dodge, J., Gururangan, S., Card, D., Schwartz, R., & Smith, N. A. (2019). Show Your Work: Improved Reporting of Experimental Results. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, 2185–2194.
- Doyle, J. (1979). A Truth Maintenance System. *Artificial Intelligence*, 12(3), 231–272.
- Dung, P. M. (1995). On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. *Artificial Intelligence*, 77(2), 321–357.
- Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130v1 (April 2024).
- Forbus, K. D., & de Kleer, J. (1993). *Building Problem Solvers*. MIT Press.

- Gao, L., Ma, X., Lin, J., & Callan, J. (2023). Precise Zero-Shot Dense Retrieval without Relevance Labels. *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1762–1777.
- Gao, T., Yen, H., Yu, J., & Chen, D. (2023). Enabling Large Language Models to Generate Text with Citations. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 6465–6488.
- Gärdenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought*. MIT Press.
- Gärdenfors, P. (2004). Conceptual Spaces as a Framework for Knowledge Representation. *Mind and Matter*, 2(2), 9–27.
- Guo, Z., Xia, L., Yu, Y., Ao, T., & Huang, C. (2025). LightRAG: Simple and Fast Retrieval-Augmented Generation. *Findings of the Association for Computational Linguistics: EMNLP 2025*, 10746–10761.
- Gupta, A., Mumick, I. S., & Subrahmanian, V. S. (1993). Maintaining Views Incrementally. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 157–166.
- He, P., Liu, X., Gao, J., & Chen, W. (2021). DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *Proceedings of the Ninth International Conference on Learning Representations (ICLR 2021)*.
- Holm, S. (1979). A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2), 65–70.
- Järvelin, K., & Kekäläinen, J. (2002). Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4), 422–446.
- Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W. (2020). Dense Passage Retrieval for Open-Domain Question Answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 6769–6781.
- Kschischang, F. R., Frey, B. J., & Loeliger, H.-A. (2001). Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2), 498–519.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474.
- Lin, J., Nogueira, R., & Yates, A. (2021). *Pretrained Transformers for Text Ranking: BERT and Beyond*. Morgan & Claypool (Synthesis Lectures on Human Language Technologies). Springer reissue 2022.
- Miracle, A. (2026). Evaluation plan for a geometric and compositional knowledge architecture [Preregistration]. GitHub repository, tag plan-v3. <https://github.com/koolamusic/geometric-knowledge-architecture-evaluation-plan>
- Moreau, L., & Missier, P. (Eds.) (2013). PROV-DM: The PROV Data Model. W3C Recommendation, 30 April 2013.

- Nogueira, R., & Cho, K. (2019). Passage Re-ranking with BERT. arXiv:1901.04085.
- Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5(1), 14–37.
- Noy, N. F., & Klein, M. (2004). Ontology Evolution: Not the Same as Schema Evolution. *Knowledge and Information Systems*, 6(4), 428–440.
- Park, J. S., O’Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 1–22. <https://doi.org/10.1145/3586183.3606763>
- Paulheim, H. (2018). How Much Is a Triple? Estimating the Cost of Knowledge Graph Creation. *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks, CEUR Workshop Proceedings 2180*. https://ceur-ws.org/Vol-2180/ISWC_2018_Outrageous_Ideas_paper_10.pdf
- Phillips, S. (2018). Going Beyond the Data as the Patching (Sheaving) of Local Knowledge. *Frontiers in Psychology*, 9, 1926. <https://doi.org/10.3389/fpsyg.2018.01926>
- Pocock, S. J. (1977). Group Sequential Methods in the Design and Analysis of Clinical Trials. *Biometrika*, 64(2), 191–199.
- Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., & Chalef, D. (2025). Zep: A Temporal Knowledge Graph Architecture for Agent Memory. arXiv:2501.13956.
- Robertson, S., & Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3(4), 333–389.
- Sakai, T. (2016). Topic Set Size Design. *Information Retrieval Journal*, 19(3), 256–283.
- Sarathi, P., Abdullah, S., Tuli, A., Khanna, S., Goldie, A., & Manning, C. D. (2024). RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024)*.
- Smucker, M. D., Allan, J., & Carterette, B. (2007). A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, 623–632.
- Terry, D. B., Demers, A. J., Petersen, K., Spreitzer, M. J., Theimer, M. M., & Welch, B. B. (1994). Session Guarantees for Weakly Consistent Replicated Data. *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, 140–149.
- Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288.
- Voorhees, E. M. (2000). The TREC-8 Question Answering Track Report. In E. M. Voorhees & D. K. Harman (Eds.), *The Eighth Text REtrieval Conference (TREC 8)* (NIST Special Publication 500-246, pp. 77–82). National Institute of Standards and Technology.
- Wang, S., Liu, Y., Xu, Y., Zhu, C., & Zeng, M. (2021). Want To Reduce Labeling Cost? GPT-3 Can Help. *Findings of the Association for Computational Linguistics: EMNLP 2021*, 4195–4205.

- Webber, W., Moffat, A., & Zobel, J. (2008). Statistical Power in Retrieval Experimentation. *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 571–580.
- Williams, A., Nangia, N., & Bowman, S. R. (2018). A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 1112–1122.
- Wittes, J., & Brittain, E. (1990). The Role of Internal Pilot Studies in Increasing the Efficiency of Clinical Trials. *Statistics in Medicine*, 9(1–2), 65–71.
- Wittes, J., Schabenberger, O., Zucker, D., Brittain, E., & Proschan, M. (1999). Internal Pilot Studies I: Type I Error Rate of the Naive t-Test. *Statistics in Medicine*, 18(24), 3481–3491.
- Zhang, C., Ré, C., Cafarella, M., De Sa, C., Ratner, A., Shin, J., Wang, F., & Wu, S. (2017). DeepDive: Declarative Knowledge Base Construction. *Communications of the ACM*, 60(5), 93–102.
- Zobel, J. (1998). How Reliable Are the Results of Large-Scale Information Retrieval Experiments? *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 307–314.