# Closed-Loop Knowledge Graph Construction: A Cost-Aware Pipeline for Streaming Business Communications

Alex Liu*
alex@korsoai.com
Korso, Inc.

Daichi Hiraoka*
daichi@korsoai.com
Korso, Inc.

Martin Pan*
martin@korsoai.com
Korso, Inc.

## Abstract

Enterprise knowledge graphs serve as memory and grounding infrastructure for AI agents, but building them from business communications is difficult: facts arrive continuously, entity mentions are ambiguous, prior context is necessary for correct extraction, and operational facts such as prices, lead times, and certifications become stale or contradictory. We study this streaming setting through a deployed manufacturing supply-chain system. Our central design decision is a *closed-loop* process: the graph being built is itself the primary input, retrieved before each extraction step and mutated only through deterministic reconciliation rules that classify each candidate fact as novel, corroborating, superseding, or contradictory. Around this loop we add a cost-ordered entity resolution cascade and predicate-specific temporal confidence maintenance, so facts with different expected lifetimes decay and supersede appropriately. Operational aggregates over an approximately two-month production window show the pipeline removes roughly half of input content volume and resolves roughly 40% of entity mentions without LLM calls. Three small pre-specified ablations quantify the design choices: on $N = 14$ hand-specified extraction scenarios, closing the loop raises strict task accuracy from 35.7% (5/14, open-loop) to 92.9% (13/14, closed-loop) at comparable model-call cost; on $N = 20$ entity mentions the cascade attains the same observed accuracy as an LLM-only baseline (18/20 for both) at 91.7% lower LLM cost; and on $N = 40$ fact pairs the deterministic reconciler assigns 39/40 four-way update decisions correctly (macro-F1 0.973) against 21/40 (macro-F1 0.384) for an append-only static-triple baseline. These are mechanism-level tests on small hand-built sets from one deployment rather than benchmark-scale accuracy estimates, and we report no confidence intervals for them.

## Keywords

knowledge graph construction, temporal knowledge graphs, entity resolution, information extraction, large language models, enterprise AI

## 1 Introduction

AI agents that operate over enterprise systems require a memory layer that is more structured than a vector index and more dynamic than a static database snapshot. Such agents must ground actions in facts about customers, suppliers, products, prices, commitments, and prior communications. In manufacturing supply chains, much of this information lives in business communications: emails, purchase orders, invoices, quote attachments, and specification sheets. These streams are noisy, multilingual, threaded, and temporally unstable. A spot price may be valid for days, payment terms may persist for months, a supplier certification may last years, and a later email may correct or supersede an earlier statement.

Large language models (LLMs) make information extraction from such streams plausible, but a direct "LLM over every document" architecture is insufficient. It is costly, repeatedly extracts similar facts, has ambiguous entity grounding, and provides little support for deciding whether a new fact should add, update, strengthen, or contradict the existing memory. For enterprise agents, these distinctions matter operationally: incorrect memory updates can drive incorrect tool use, while stale memory can make an otherwise capable agent act on obsolete prices, lead times, or supplier constraints.

We argue that enterprise graph construction should be treated as a streaming ML systems problem. The objective is not only extraction accuracy on isolated documents, but the joint behavior of extraction, entity grounding, model-call cost, temporal validity, and auditability as documents arrive over time. This framing changes the system design. Prior graph facts become useful extraction context. Entity resolution should escalate from cheap deterministic tests to expensive model calls only when needed. Fact confidence should depend on predicate semantics and age. Graph mutation should be governed by auditable reconciliation rules rather than unconstrained model output.

We study these principles in a deployed manufacturing supply-chain setting. The system processes multilingual business communications and constructs a temporal knowledge graph through seven stages: compression, classification, extraction, entity resolution, reference resolution, synthesis, and reconciliation. The deployment used a relational PostgreSQL backend, preserved provenance from source document to graph write, and used LLMs as one component in a broader cost-aware pipeline rather than as the sole mechanism.

This paper makes three contributions.

(1) **Closed-loop graph construction** (Sections 4.4 and 4.7). We design and deploy a pipeline in which the graph being built is the primary input to its own construction: prior facts are retrieved before each extraction, and graph mutation occurs only through deterministic reconciliation rules that classify each candidate fact as novel, corroborating, superseding, or contradicting. The contribution is a concrete production instantiation of construction-time retrieval coupled with predicate-cardinality-aware reconciliation for streaming business documents.

(2) **Cost-aware components around the loop** (Sections 4.5 and 4.7). We present the surrounding pipeline elements that make the closed loop practical at production cost: a five-tier cost-ordered entity resolution cascade and predicate-specific temporal confidence maintenance with closed-form half-life decay.

*All authors contributed equally as first authors.

(3) **Operational evidence and ablations** (Sections 5 and 6). We report deployment aggregates and three mechanism-level ablations covering extraction context, entity-resolution cost, and temporal reconciliation. The reconciliation rules and the cascade are evaluated directly; the half-life decay parameterization is not, and Section 8 says so explicitly.

## 2 Related work

*LLM-based information extraction and knowledge graph construction.* LLMs have been used for relation extraction, schema canonicalization, and open-domain knowledge graph construction [6, 8, 14, 18, 19]. Most pipelines focus on extracting triples from isolated documents or on canonicalizing extracted relations after the fact. Our setting is different: each new communication is part of a stream and must be interpreted relative to prior business state. This makes update semantics, provenance, and temporal validity first-class design requirements.

*Retrieval-augmented systems and agent memory.* GraphRAG-style systems build entity graphs so that downstream retrieval and summarization can be improved [5]. Temporal agent-memory systems such as Graphiti [12] and A-Mem [16], and benchmarks like LongMemEval [15], treat memory as an evolving artifact for downstream agents and study how it is queried and maintained over time: the graph informs the consumer, not the constructor. The point of departure here is that the graph informs *its own construction*: prior facts are retrieved before each extraction and used to interpret the new document, decide whether the candidate updates an existing fact, and select the reconciliation branch. Construction-time retrieval and cardinality-aware reconciliation together close the loop in a way absent from prior agent-memory pipelines.

*Temporal knowledge graphs.* Research on temporal knowledge graphs has developed embedding and reasoning models over time-stamped graph snapshots [3, 4, 7, 9]. Those methods typically assume a graph exists and evaluate temporal reasoning or outdated-fact filtering. Our focus is the upstream operational problem: how to construct an evolving graph from business documents while representing freshness, supersession, contradiction, and reviewability.

*Entity resolution and cost-aware cascades.* Entity resolution (ER) is often described as blocking, matching, and merging [2]. Recent work studies LLMs for entity matching [11] and cost-efficient prompting [10]. Our pipeline applies a FrugalGPT-style cascade [1] to entity resolution: exact aliases, deterministic identifiers, lexical similarity, embedding similarity, and LLM disambiguation are invoked in increasing cost order. This is not merely an optimization; it changes reliability because deterministic tiers can be audited and cached.

*Production ML systems.* ML systems often fail because of hidden dependencies, feedback loops, configuration debt, and monitoring gaps [13]. Compound AI systems combine retrieval, tools, models, and control logic rather than relying on a single model call [17]. The system studied here is a concrete instance of that pattern in enterprise graph construction: model calls are embedded inside a pipeline with deterministic gates, telemetry, confidence maintenance, and human review.

## 3 Problem formulation

Let $D_t$ denote the document or message arriving at time $t$, and let $G_{t-1}$ denote the graph state before processing it. The system must produce candidate facts

$$F_t = \{(s, p, o, \tau, c, e)\},$$

where $s$ and $o$ are resolved or unresolved entities, $p$ is a predicate from a controlled ontology, $\tau$ is temporal validity metadata, $c$ is confidence, and $e$ is provenance linking the fact to source evidence. Unlike static extraction, the desired output depends on both the current document and prior graph state:

$$F_t = f(D_t, G_{t-1}, C_t),$$

where $C_t$ contains thread, tenant, sender, and artifact context.

This setting induces four requirements. **Incrementality:** documents arrive continuously and cannot be processed as an independent batch. **Grounded entity resolution:** entity mentions must map to tenant-specific products, companies, contacts, and documents. **Temporal validity:** fact confidence should depend on age and predicate semantics. **Auditable mutation:** the system must distinguish corroboration, supersession, contradiction, and novelty in a way that operators can inspect.

These requirements imply a different objective from ordinary document-level extraction. Let $q(F_t)$ denote the quality of the facts committed from document $D_t$, measured by correctness, grounding quality, temporal validity, and duplicate avoidance; let $m(F_t, G_{t-1})$ denote mutation risk, such as overwriting a still-valid fact; and let $k(D_t)$ denote processing cost and latency. The operational objective is not simply

$$\max \mathbb{E}[q(F_t)],$$

but rather to maximize retained graph utility under cost and mutation-risk constraints:

$$\max \mathbb{E}[q(F_t) - \lambda_m m(F_t, G_{t-1}) - \lambda_k k(D_t)].$$

The constants are deployment-specific, but the decomposition is useful because it separates three failure classes that are often conflated. A system may extract correct facts but attach them to the wrong product; it may extract and ground correctly but leave a stale prior fact active; or it may do both correctly while spending too much on model calls to be practical for continuous mailbox processing.

The formulation also clarifies the role of the graph. The graph is not only the final database populated by extraction. It is part of the input to the next extraction step, the state against which new facts are reconciled, and the evidence base for downstream agents. This makes construction a closed-loop process: errors in $G_{t-1}$ can influence $F_t$, while correct updates in $F_t$ improve future extraction. Accordingly, we design each stage to expose confidence, provenance, and fallback behavior rather than returning only final triples.

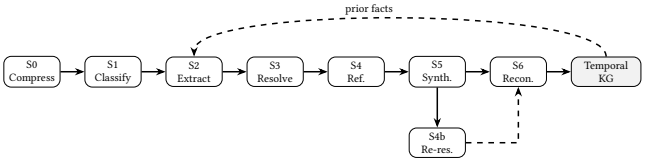


**Figure 1: Seven-stage construction pipeline. Prior facts are retrieved from the graph before extraction, making graph construction a closed-loop process rather than independent document extraction.**

# 4 System design

## 4.1 Pipeline overview

Figure 1 summarizes the deployed pipeline. Stage 0 compresses input, removes boilerplate, preserves structured tables, and reconstructs thread context. Stage 1 classifies documents into business types and speech acts using a cost pyramid. Stage 2 extracts atomic facts, conditioned on retrieved prior graph facts. Stage 3 resolves entity mentions. Stage 4 resolves vague references such as "same specifications as before." Stage 5 synthesizes facts across related documents. Stage 6 reconciles each incoming fact against existing graph state.

The implementation maintained provenance through all stages. This is important because the graph is used as agent memory: facts must be auditable back to source documents if they later affect a tool call, customer-facing response, or human review decision.

## 4.2 Compression and business-routing front end

The first two stages are deliberately conservative. Business mailboxes contain repeated signatures, legal footers, quoted replies, automated notifications, and multi-message threads where the newest content is small relative to the full body. Processing the raw text with an LLM wastes context and cost, and can cause older quoted text to be misread as a new commitment. Stage 0 therefore performs exact deduplication using content hashes, near-deduplication using sliding-window overlap, boilerplate removal, and HTML-table preservation. Preserving tables before stripping markup is important because prices, quantities, delivery windows, and item specifications often appear only as table rows.

Stage 1 routes messages by both document type and speech act. For example, a purchase order, invoice, quote revision, specification sheet, payment reminder, and shipping update have different expected predicates and different risk profiles. Similarly, a message can request a quote, revise a prior quote, confirm an order, reject a proposal, or provide missing documentation. The routing stage uses cheap rules for obvious low-value messages, embedding prototypes for common classes, and an LLM only for ambiguous cases. This stage does not write to the graph; it determines which schema constraints, retrieval scopes, and extraction prompts should be used downstream.

## 4.3 Atomic facts and predicate control

The central data structure is an atomic fact:

$$f = \langle s, p, o, [t_s, t_e], c, prov \rangle.$$

Here $[t_s, t_e]$ is the real-world validity interval, while system timestamps track creation, update, and expiration inside the graph. Predicates are drawn from a bounded ontology rather than generated freely. Representative predicates include supplier relationships, contact-company relations, quoted prices, lead times, payment terms, certifications, and product specifications.

Open predicate extraction produced many near-duplicate relation labels during development. The controlled ontology prevents this predicate explosion and routes facts to appropriate storage paths. For example, pricing observations, company attributes, graph relationships, and temporal episodes have different query and lifecycle requirements. Facts with non-canonical predicates are normalized through prompt constraints, tense normalization, and post-extraction hard filters.

We found it useful to separate *relation facts* from *observation facts*. A relation fact, such as `supplies(company, product)`, describes a relatively durable edge that can be corroborated over time. An observation fact, such as a quoted price or lead time, is often valid only in the context of a supplier, product, customer, date, currency, quantity break, and source document. Conflating these into a single triple table makes downstream agent behavior brittle: a current quote and an old quote look like competing objects of the same predicate even when both should be retained for audit. The storage layer therefore demultiplexes extracted facts into relationship edges, pricing observations, company knowledge, temporal episodes, contradiction queues, and unresolved-fact queues.

## 4.4 Prior-memory retrieval

Before extraction, the system retrieves relevant prior facts from $G_{t-1}$. The retrieval scope uses sender/counterparty identity, thread context, semantic similarity, and recency. The retrieved facts are injected into the extraction prompt in structured form. This allows the extractor to interpret referential language, detect updates, and avoid repeatedly creating duplicate facts.

The feedback loop is the main conceptual difference from ordinary retrieval-augmented generation. Retrieval is not used only to answer a query; it is used to update the memory that will condition future extraction. In deployment, this was most useful for quote updates, recurring product references, and negotiation threads where later messages modify earlier facts rather than restating complete information.

The retrieval result is intentionally structured rather than free-form. Each prior fact is passed with subject, predicate, object, confidence, validity interval, source timestamp, and source identifier. This prevents the extractor from treating retrieved summaries as ungrounded natural-language context. It also supports prompt-level instructions such as: create a new fact only when the current document adds evidence; mark a fact as an update when it supersedes a retrieved fact; and leave a fact unresolved if the current document requires an entity not present in retrieved context. In practice, this design reduces the amount of context needed because the model

receives the small set of candidate prior facts instead of a long thread transcript.

## 4.5 Cost-ordered entity resolution

Entity mentions are resolved through a five-tier cascade. The first tier uses exact aliases and canonical names. The second uses deterministic manufacturing identifiers such as part numbers and structured specifications. The third uses fuzzy lexical similarity. The fourth uses embedding similarity or a composite matcher. The final tier uses LLM disambiguation for unresolved or ambiguous cases. Each tier is invoked only if earlier tiers fail or return insufficient confidence.

Structured products illustrate why pure semantic similarity is insufficient. Two products that differ only in thickness can be semantically close but operationally distinct. For product candidates, the system therefore combines grade match, finish match, dimension match, and name similarity:

$$\text{score} = 0.35g + 0.15f + 0.35d + 0.15n.$$

This gives structured identifiers more weight than generic name similarity.

The cascade has two roles. First, it reduces cost by resolving common mentions without an LLM. Second, it creates interpretable failure modes. If an exact alias tier resolves incorrectly, the alias table is wrong; if the dimension matcher resolves incorrectly, the structured specification parser is wrong; if an LLM tier resolves incorrectly, the prompt or retrieved candidate set is wrong. These distinctions matter for maintenance because entity resolution errors can poison the graph more severely than missing facts: a wrong merge can cause future messages about one supplier or product to update another.

## 4.6 Reference resolution and cross-document synthesis

Business communications frequently use anaphora and implicit references. Messages such as “same as last time,” “use the revised price,” or “the attached spec supersedes the old one” cannot be interpreted from the current body alone. Stage 4 resolves these references using three escalating scopes. The cheapest scope is the thread card: recent participants, referenced documents, known products, and prior commitments in the same conversation. The second scope uses hybrid retrieval over graph facts and document metadata. The most expensive scope queries the broader graph when the thread does not contain enough evidence. Unsupported LLM reference claims are capped at low confidence and routed to unresolved review rather than written directly.

Stage 5 synthesizes facts across related artifacts. For example, a customer email may request a product, an attached spreadsheet may contain quantities, and a later invoice may confirm price and shipment terms. Synthesis batches related artifacts into a single episode when the evidence indicates one business event. This stage is separated from extraction because document-level facts and episode-level facts have different provenance: the former point to a single artifact, while the latter depend on a bundle of evidence.

## 4.7 Temporal confidence and deterministic reconciliation

Facts receive predicate-specific temporal expectations. Spot prices decay quickly; supplier relationships and product specifications decay more slowly. The deployed system attenuated fact confidence by a closed-form exponential decay whose predicate-specific half-lives ranged from 7 days for spot prices to 1,825 days for company attributes and product specifications, with a confidence floor of 0.3 so facts remain available for audit rather than silently disappearing. These half-lives were set by domain expert consultation rather than learned, and we evaluate the reconciliation rules below but not the decay parameterization.

Incoming facts are reconciled deterministically. If no prior fact exists for the subject-predicate pair, the fact is novel. If the same target is observed again, the fact corroborates prior memory. If the predicate is single-valued and the incoming fact temporally overlaps a different target, the system either supersedes the old fact or routes a contradiction to review. Multi-valued predicates skip contradiction checks. This makes graph mutation inspectable and avoids allowing unconstrained model output to overwrite high-impact memory.

Reconciliation uses both predicate cardinality and temporal overlap. For single-valued predicates, such as a current preferred supplier for a specific product under a tenant, an incoming conflicting target cannot simply be appended as another active value. For multi-valued predicates, such as possible suppliers or contacts associated with a company, a different target may be valid and should not trigger a contradiction. For observation predicates, such as quoted prices, apparent conflicts may be valid if quantity, currency, customer, or effective date differ. The reconciliation layer therefore normalizes comparison keys before applying cardinality rules.

Temporal confidence is maintained separately from support count. A fact can have high historical support and low current confidence if its predicate decays quickly. Conversely, a newly observed fact can have modest support but high current confidence when it comes from a high-quality source and matches a predicate with short expected lifetime. This distinction is important for agents: an old price should remain visible for audit and trend analysis, but should not be treated as equally actionable for a new quote.

## 4.8 Operational safeguards

The deployment used several safeguards that are relevant to ML systems operating over private business data. Unresolved entities and contradictions are queued rather than silently dropped. All graph mutations preserve source evidence. LLM calls are isolated behind cheaper deterministic and embedding stages. Circuit breakers and queue isolation limit failures in model-backed stages. Company-level purge gates drain in-flight graph jobs before destructive operations. These controls are part of the scientific object under study: they determine whether an extracted graph can function as reliable agent memory rather than an untrusted byproduct of document processing.

The safeguards also define evaluation boundaries. If a system routes contradictions to review, contradiction recall and review load both matter. If a system caps unsupported reference claims,

**Table 1: Retained cost-routing aggregates from the deployed system, measured over an approximately two-month production window. Percentages are shares of input content volume (compression) or of entity mentions (free resolution share); the routing split is the share of Stage 1 decisions taken at each tier. These values describe how work is distributed across deterministic, embedding, and LLM tiers; they are not a substitute for held-out accuracy evaluation, which is treated separately in the replay protocol.**

| Operational property | Observed aggregate |
|---|---|
| Deployment window | ∼2 months |
| Pipeline implementation footprint | ∼60,000 LOC across 124 service files |
| Supported languages | EN, JA, ZH, KO |
| Compression effect | ∼50% content-volume reduction |
| Free entity-resolution share | ∼40% at zero LLM cost |
| Stage 1 routing split | 15% / 35% / 50% heuristic / embedding / LLM |

some true facts may be delayed, but false graph writes should decrease. If a system uses deterministic tiers before LLM tiers, the right metric is not only end-to-end accuracy but also the cost and error profile of each tier. We therefore treat the deployed system as a compound ML system whose components must be evaluated jointly and individually.

## 5 Operational evidence

The deployment provides evidence that the system was implemented and operated at realistic scale. The pipeline supported multilingual English, Japanese, Chinese, and Korean communications; used a controlled predicate ontology; and persisted provenance, quality, and stage telemetry. Retained aggregates over an approximately two-month production window show that compression removes about 50% of input content volume and that the cost-ordered entity resolution cascade resolves about 40% of mentions at a zero-LLM tier. Table 1 collects these aggregates, Table 2 records where model calls occur across the seven stages, and Table 3 lists the recurring failure modes that motivated the architecture.

The evidence in Tables 1–3 supports two limited conclusions. First, the architecture was implemented beyond a toy prototype and exercised on realistic multilingual document streams. Second, cost-aware routing is operationally meaningful: a substantial share of work can be handled before expensive LLM calls, and the stages that do use models can be isolated behind confidence gates. The evidence does not by itself establish extraction accuracy, entity-resolution accuracy, or contradiction precision; those require the replay protocol below.

## 6 Evaluation: ablations and framework

The operational evidence shows the system was deployed; the key scientific question is which design components drive the cost-quality tradeoff. We approach this in two parts: (i) three executed mechanism-level ablations directly testing the closed-loop, cascade, and reconciliation claims, and (ii) a replay-protocol framework specifying the larger labeled evaluation needed for deployment-scale accuracy estimates. The success criterion throughout is a Pareto improvement on the *construction operating frontier*: for a given model-call cost and review-load budget, fewer duplicate or stale facts than append-only extraction, lower model-call cost than LLM-only resolution, and fewer unsafe graph mutations than unconstrained model-written memory. The three executed ablations address the model-call-cost and update-decision legs of this criterion; duplicate-fact rate and unsafe-mutation rate belong to the

**Table 2: Qualitative LLM-call profile, primary output, and cost-control mechanism for each of the seven pipeline stages; no numeric measurement is reported here. Model calls are concentrated where deterministic or embedding-based mechanisms are insufficient. Cost refers to LLM calls only, excluding database, embedding, human-review, and engineering costs. Reconciliation is intentionally LLM-free because it governs committed graph state.**

| Stage | LLM-call profile | Primary output | Cost-control mechanism |
|---|---|---|---|
| Compress | none | Reduced document; tables preserved | Deduplication; boilerplate stripping; thread reconstruction |
| Classify | none to low | Type, speech act, retrieval hints | Heuristic and embedding tiers before LLM routing |
| Extract | low to medium | Atomic candidate facts | Flash-first extraction with confidence and utilization gates |
| Resolve | none to low | Grounded entities | Deterministic and embedding tiers before LLM disambiguation |
| Ref. resolve | none to low | Linked prior facts and artifacts | Thread card and retrieval before full model fallback |
| Synthesize | low to medium | Cross-document episodes | Deferred batch processing and quality fallback |
| Reconcile | none | Graph mutations or review items | Deterministic temporal/cardinality rules |

**Table 3: Six failure modes observed during deployment that motivated the final architecture, each paired with its cause in streaming input and the corresponding system response; the table is qualitative and reports no frequencies. The system treats these as expected operating conditions rather than exceptional errors.**

| Failure mode | Why it appears in streams | System response |
|---|---|---|
| Stale operational facts | Prices, lead times, and availability change without invalidating all history | Predicate half-lives, current-confidence queries, retained provenance |
| Alias-heavy ambiguity | Suppliers and products are referred to by abbreviations, local names, or part numbers | Cascaded entity resolution with deterministic identifiers before LLM fallback |
| Single-valued conflicts | Later communications revise an earlier commitment or identify a mistake | Cardinality-aware supersession or contradiction review |
| Model queue saturation | Continuous ingestion can exceed expensive model capacity | Cost pyramid, confidence gates, deferred synthesis, isolated queues |
| Thread fragmentation | Forwarded or copied messages break normal conversation lineage | Thread cards plus hybrid graph/document retrieval |
| Unsupported reference claims | A model may infer that "same as before" points to the wrong prior artifact | Evidence gates and confidence caps for unresolved references |

**Table 4: Prior-memory retrieval ablation on the hand-specified $N = 14$ scenario set. Rows report counts with percentages, mean tokens per scenario, and mean LLM cost per scenario in USD. Closed-loop extraction is more accurate at comparable model-call cost; the single remaining closed-loop failure is a contradiction label that Stage 6 routes to human review.**

| Metric | Open-loop | Closed-loop |
|---|---|---|
| Strict task accuracy | 5 / 14 (35.7%) | 13 / 14 (92.9%) |
| Decision-label correctness | 7 / 14 (50.0%) | 13 / 14 (92.9%) |
| Mean prompt tokens / scen. | 336 | 393 |
| Mean completion tokens / scen. | 160 | 138 |
| Mean LLM cost / scen. (USD) | $5.0\times10^{-4}$ | $4.6\times10^{-4}$ |
| Cost change vs. open-loop | (reference) | −7% |

held-out replay protocol and are not yet measured. Because production communications are private, the executed ablations run on hand-specified synthetic inputs rather than customer data; Appendix D states precisely which parts of that harness are and are not currently available.

## 6.1 Executed ablations

We executed all three planned ablations. The first two were run against `gemini-2.5-flash` at temperature 0; the third is purely deterministic and exercises the LLM-free Stage 6 reconciler. The scenarios were specified before scoring, every scenario output was scored, and no failed case was excluded from the reported results.

*Prior-memory retrieval ($N = 14$).* We hand-built a test set of $N = 14$ B2B email scenarios: eight *cooperative* (three quote-updates, two duplicate-fact reminders, two reference-resolution, one explicit contradiction) where prior-fact retrieval is expected to help, plus six *adversarial* ones where prior context is stale, refers to a similar-but-distinct entity, is irrelevant to the email, or sets up a recency or predicate trap. For every scenario the prior-fact set is hand-specified, so the expected decision label and resolved object are known by construction. Each scenario was run twice: OPEN-LOOP (email only) and CLOSED-LOOP (email plus a structured prior-fact block injected exactly as the production retrieval stage would inject it). Scoring was exact-match on the decision label and on presence of the expected resolved object; Table 4 reports the result.

*Cost-aware entity resolution ($N = 20$).* We then ablated the five-tier ER cascade against an LLM-only baseline on $N = 20$ entity mentions resolved against a fixed catalog of 23 entities (12 companies, 11 products). Mention forms span exact aliases, deterministic specification strings, lexical variants and typos, semantic paraphrases, and two genuinely unresolvable mentions whose gold label is NONE. The cascade invokes deterministic, lexical, and similarity tiers before any LLM call; in this synthetic setting the embedding tier is represented by a deterministic composite-similarity proxy, so the test runs without embedding credentials. The LLM-only variant sends every mention to the model with the full candidate catalog; Table 5 reports the result.

**Table 5: Entity-resolution cascade vs. LLM-only resolution on $N = 20$ mentions against a fixed 23-entity catalog. Rows report counts and LLM cost in USD. On this test set the cascade attains the same observed accuracy as LLM-only resolution at 91.7% lower total LLM cost, escalating only 3/20 mentions to the model.**

| Metric | Cascade | LLM-only |
|---|---|---|
| Resolution accuracy | 18 / 20 (90%) | 18 / 20 (90%) |
| LLM calls per 20 mentions | 3 | 20 |
| Total LLM cost (USD) | $2.1\times10^{-4}$ | $25.2\times10^{-4}$ |
| Cost per mention (USD) | $1.0\times10^{-5}$ | $1.3\times10^{-4}$ |
| Cost change vs. LLM-only | −91.7% | (reference) |

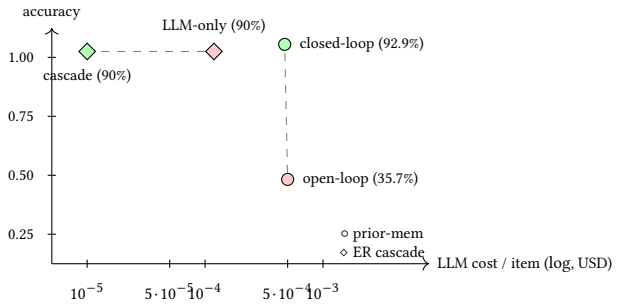


**Figure 2: Cost–accuracy frontier from the two LLM-driven ablations ($N = 14$ scenarios and $N = 20$ mentions). Closed-loop extraction improves accuracy at comparable cost; cascaded entity resolution obtains the same observed accuracy as LLM-only resolution on this test set at 91.7% lower total cost.**

*Temporal reconciliation ($N = 40$ fact pairs).* We evaluated the deterministic Stage 6 reconciler against an append-only static-triple baseline on $N = 40$ synthetic fact pairs: eight hand-built edge cases plus a 32-case grid spanning the four decision categories and exercising overlap, non-overlap, recency, and predicate cardinality. The full bi-temporal cardinality-aware rule reaches 39/40 exact-match on the four-way decision label (macro-F1 0.973) versus 21/40 (macro-F1 0.384) for the baseline; it recovers all 10/10 supersession cases and 8/9 contradiction cases, while the static-triple baseline recovers none of either class. The one miss is the ambiguous contradiction edge case in which the rule applied SUPERSEDES instead of CONTRADICTS; we report it here as a documented failure case rather than excluding it.

*Reading the three ablations together.* Each ablation supplies one piece of evidence per design claim: prior-memory retrieval along the accuracy axis (92.9% vs. 35.7% at comparable cost), the ER cascade along the cost axis (91.7% cheaper at the same observed accuracy on this test set), and Stage 6 reconciliation along the four-way-decision axis (macro-F1 0.973 vs. 0.384, all supersession cases and 8/9 contradiction cases recovered). Figure 2 plots the two LLM-driven ablations on the cost–accuracy plane. These are mechanism tests, not benchmark-scale accuracy estimates; the full replay protocol with held-out gold labels and bootstrap confidence intervals is specified in Appendix D.

## 7 Discussion

The system illustrates a broader lesson for agent memory: better extraction models alone do not solve the operational memory problem; memory construction requires explicit decisions about when to retrieve prior context, when to call expensive models, how to ground ambiguous entities, and how to represent stale or conflicting facts, and these decisions change the behavior of the resulting agent. The strongest generalizable pattern is the separation between model-backed proposal and deterministic mutation: LLMs and embeddings are useful for extracting and resolving candidate facts, but graph mutation is safer when governed by predicate cardinality, temporal overlap, provenance, and explicit review queues. This separation is also what makes the system evaluable: extraction, entity resolution, reference resolution, and reconciliation can be ablated without changing graph-write semantics.

## 8 Limitations and ethics

The deployment is single-domain; the closed-loop construction pattern and cardinality-aware reconciliation should transfer, but the predicate ontology and per-predicate half-lives reflect manufacturing-supply-chain conventions and need re-derivation elsewhere, and a complete accuracy study still requires held-out annotation or replay with anonymized artifacts. The mechanism-level ablations run on synthetic inputs, which does not replace a held-out replay evaluation over anonymized production-like document bundles; further, as Appendix D records, the per-run outputs behind the three reported mechanism results are not currently available for release, so those results cannot at present be independently recomputed. The three executed ablations are small and hand-constructed: they test whether each mechanism does what it is designed to do, and they do not establish deployment-scale accuracy. The half-life decay parameters were set through expert consultation rather than learned from data, and are therefore potentially sub-optimal; the temporal-reconciliation ablation validates the cardinality and overlap logic but not the decay parameterization. No statistical uncertainty is reported, because the final held-out evaluation and replay experiments have not been completed. The underlying communications may contain commercially sensitive data, so public artifacts must be synthetic or anonymized; human review remains necessary for high-impact contradictions and low-confidence resolutions, and provenance, review queues, confidence decay, and deterministic reconciliation mitigate but do not eliminate downstream-agent harm.

## 9 Conclusion

This paper frames knowledge graph construction for enterprise agents as streaming, cost-aware memory construction rather than isolated document extraction. The central finding is that the graph should participate in its own construction: prior facts retrieved before extraction improve update and reference handling, while committed graph state is protected by deterministic temporal and cardinality-aware reconciliation. In the deployed manufacturing setting, compression removed roughly half of input volume and the entity-resolution cascade handled roughly 40% of mentions at zero LLM cost. Mechanism-level ablations support the same design story: closed-loop extraction improves strict task accuracy from 35.7% (5/14) to 92.9% (13/14), cascaded entity resolution obtains the same observed 18/20 accuracy as LLM-only resolution at 91.7% lower LLM cost, and deterministic reconciliation improves four-way update decisions from 21/40 (macro-F1 0.384) to 39/40 (macro-F1 0.973). These results come from three small hand-built test sets in a single deployment, so they should be read as evidence that each mechanism behaves as designed rather than as generalizable accuracy estimates. The broader claim is that construction-time retrieval, cost-ordered grounding, temporal confidence, and deterministic mutation are separable, measurable design choices for auditable agent memory from noisy business streams.

## A Pipeline details

Stage 0 performs three-layer deduplication: exact hash matching, sliding-window near-deduplication, and boilerplate stripping. It also converts HTML tables to tabular text before stripping markup, which is important because prices and line items frequently appear in email tables. Thread reconstruction uses parent identifiers and chronological ordering to build conversation context.

Stage 1 uses a cost pyramid. Heuristics catch low-value or obvious messages such as automated notifications. Embedding prototypes handle common business document types. LLM classification is reserved for ambiguous messages and produces document type, speech act, and reference hints for downstream retrieval.

Stage 2 follows a flash-first extraction design. An initial low-cost model proposes facts. Confidence gates decide whether escalation is necessary. If confidence is low or pricing-critical fields are missing, bounded rescue retrieves additional context and reruns extraction.

## B Predicate and storage design

Table 6 lists representative predicates from the controlled ontology of Section 4.

**Table 6: Eight representative predicates from the controlled ontology, with the subject and object types each accepts and the storage route it is written to. The table is illustrative and does not enumerate the full deployed predicate set.**

| Predicate | Subject | Object | Route |
|---|---|---|---|
| `supplies` | company | company | relationship |
| `buys_from` | company | company | relationship |
| `works_for` | contact | company | relationship |
| `manufactures` | company | product | relationship |
| `quoted_price` | company | product | pricing |
| `cost_price` | company | product | pricing |
| `has_lead_time` | company/product | none | knowledge |
| `negotiation_for` | company | product | episode |

Facts are demultiplexed into storage routes rather than written to a single generic triple table. This trades uniformity for operational fit: pricing observations, relationship edges, company attributes, and temporal episodes have different query patterns, access policies, and retention needs.

## C Reconciliation decision procedure

For an incoming fact $f = \langle s, p, o, [t_s, t_e], c \rangle$, reconciliation proceeds as follows. If $s$ is unresolved, the fact is queued or marked novel with low confidence. If there are no active prior facts for $(s, p)$, the fact is novel. If an active prior fact has the same target, the fact corroborates memory and increments support. If $p$ is multi-valued, a different target is also novel. If $p$ is single-valued and time intervals overlap, the newer fact supersedes the older fact when recency and evidence support the update; otherwise it is routed as a contradiction. This procedure is deterministic and does not call an LLM.

## D Replay protocol and reproducibility

Because the deployment processes private business communications, raw data cannot be released; reproducibility relies on a replay protocol against anonymized or synthetic surrogates that preserve the decision structure the pipeline must solve.

*Evaluation unit.* The evaluation unit is a thread or document bundle, not an isolated message: chronologically ordered communications, extracted attachments, a tenant entity catalog snapshot, and the graph state preceding the first message in the bundle. Raw communications can be replaced by anonymized surrogates as long as predicates, entity ambiguity, table structure, language, and update relationships are preserved; the anonymization target is decision-structure preservation, not lexical realism.

*Gold labels.* Labels cover document type and speech act; extracted atomic facts; canonical subject and object entities; and the per-fact reconciliation decision (novel, corroborating, superseding, contradictory, or review-only). Pricing and lead-time labels additionally include quantity, currency, unit, product, supplier, and effective-date fields, since these often distinguish true contradictions from compatible observations. Entity-resolution labels distinguish unresolved-but-safe cases from incorrect merges; the latter are more harmful to memory quality.

*Baselines.* Four baselines isolate the design choices: **document-only extraction** runs the extractor without prior graph retrieval; **static-triple memory** keeps the same extractor and resolver but removes validity intervals, confidence decay, and reconciliation decisions; **LLM-only resolution** sends every ambiguous mention to the highest-cost tier; **deterministic-only resolution** disables embedding and LLM disambiguation. The two ablations executed in §6.1 compare against the first and third of these.

*Statistical treatment.* Confidence intervals over held-out annotations are computed by bootstrapping at the natural unit (pipeline job for cost/latency, fact or fact pair for extraction quality and contradiction outcomes); thread-level grouping respects within-thread dependence.

*Artifact availability.* No artifact accompanies this preprint: the three mechanism-test values have no releasable provenance record, and we make no reproducibility claim for them. The harness that produced them is synthetic, with `run_ablation.py` and `run_er_ablation.py` defining the prior-memory scenarios and the entity catalog and mention set and calling a compatible model API, and `run_recon_ablation.py` running the deterministic Stage 6 reconciler over hand-specified fact pairs without credentials. Its outputs, however, are missing: the per-run model-output JSON behind Tables 4–5 and behind the reconciliation result, a checksum manifest over it, and a complete scoring implementation are unavailable; the scoring module we hold has correct pricing constants but unimplemented match predicates; and the reconciliation harness we hold enumerates an earlier, smaller fact-pair set rather than the $N = 40$ set reported here. Any future release would have to ship scenario definitions, gold labels, per-run outputs, and a working scorer before a reproducibility claim about these numbers would be warranted. The design contribution does not depend on that record: the graph mutation rules, cascade order, and decay parameters are deterministic and fully specified in Sections 4.5 and 4.7 and Appendix C.

## GenAI usage disclosure

Large language models were used for limited drafting, editing, and format-conversion assistance. LLMs are also one component of the system studied in the paper, where they are used for information extraction and disambiguation after lower-cost deterministic and embedding-based stages. We are responsible for the manuscript content, citations, reported empirical values, and all claims.

## References


[1] L. Chen, M. Zaharia, and J. Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

[2] V. Christophides, V. Efthymiou, T. Palpanas, G. Papadakis, and K. Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys*, 53(6):127:1–127:42, 2020.

[3] S. S. Dasgupta, S. N. Ray, and P. Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proc. EMNLP*, pages 2001–2011, 2018.

[4] F. Ding, T. Wang, Y. Gao, S. Yu, J. Ren, and F. Xia. HALO: Half life-based outdated fact filtering in temporal knowledge graphs. *arXiv:2505.07509*, 2025.

[5] D. Edge et al. From local to global: A graph RAG approach to query-focused summarization. *arXiv:2404.16130*, 2024.

[6] P. Huguet Cabot and R. Navigli. REBEL: Relation extraction by end-to-end language generation. In *Findings of EMNLP*, pages 2370–2381, 2021.

[7] W. Jin, M. Qu, X. Jin, and X. Ren. Recurrent event network: Autoregressive structure inference over temporal knowledge graphs. In *Proc. EMNLP*, pages 6669–6683, 2020.

[8] M. Josifoski, N. De Cao, M. Peyrard, F. Petroni, and R. West. GenIE: Generative information extraction. In *Proc. NAACL*, pages 4626–4643, 2022.

[9] T. Lacroix, G. Obozinski, and N. Usunier. Tensor decompositions for temporal knowledge base completion. In *Proc. ICLR*, 2020.

[10] N. Nananukul, K. Sisaengsuwanchai, and M. Kejriwal. Cost-efficient prompt engineering for unsupervised entity resolution in the product matching domain. *Discover Artificial Intelligence*, 4(1), 2024.

[11] R. Peeters and C. Bizer. Entity matching using large language models. In *Proc. EDBT*, 2025.

[12] P. Rasmussen et al. Zep: A temporal knowledge graph architecture for agent memory. *arXiv:2501.13956*, 2025.

[13] D. Sculley et al. Hidden technical debt in machine learning systems. In *Proc. NeurIPS*, pages 2503–2511, 2015.

[14] S. Wadhwa, S. Amir, and B. C. Wallace. Revisiting relation extraction in the era of large language models. In *Proc. ACL*, pages 15566–15589, 2023.

[15] D. Wu et al. LongMemEval: Benchmarking chat assistants on long-term interactive memory. In *Proc. ICLR*, 2025.

[16] W. Xu et al. A-MEM: Agentic memory for LLM agents. In *Proc. NeurIPS*, 2025.

[17] M. Zaharia et al. The shift from models to compound AI systems. *BAIR Blog*, 2024.

[18] B. Zhang and H. Soh. Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. In *Proc. EMNLP*, 2024.

[19] Y. Zhu et al. LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27, Article 58, 2024.