

Gold data for agents: an engineering field guide

2026-10-04

Abstract. Agent quality is bounded by context quality. An engineering field guide to building a governed, versioned, multi-representation gold data layer: ingestion, parsing, conformance, gold artifact design, retrieval, MCP serving, context engineering, evaluation, operations, and security.

0. Executive Summary

Thesis. Agent quality is bounded by context quality. Most production agent failures trace to retrieval and data problems rather than model capability: the context was missing, stale, contradictory, unattributable, or drowned in noise. Anthropic’s guidance frames this as *context engineering*: curating the smallest set of high-signal tokens that maximizes the probability of the desired outcome [A3].

Core claims of this guide:

1. **Medallion (bronze/silver/gold) is the right ingestion discipline but the wrong definition of “done.”** Warehouse gold is optimized for aggregates and dashboards. Agent gold must be optimized for *retrieval precision, attribution, non-contradiction, and actionability*.
2. **Gold for agents is multi-representational.** The same truth is served as canonical documents, atomic claims, entity graphs, a semantic layer over structured data, procedures/skills, and tools. Different query types need different representations.
3. **Build the evaluation set before the pipeline.** A gold layer without a measured retrieval and task-success baseline is unfalsifiable.
4. **MCP is the interface layer, not the storage layer.** Model Context Protocol standardizes how agents discover and call your gold layer. It does not make bad data good.
5. **Security is a data property.** Permissions, provenance, and injection resistance must be captured at ingest and propagated to every index. Bolting them on at serving time leaks data.

The scarce resource

Not tokens. The scarce resource is **non-contradictory, attributable, procedure-complete context**. Everything in this guide is in service of producing it reliably and proving that you have.

1. Definitions: What “Gold” Means for Agents

1.1 Medallion architecture (origin)

Popularized by Databricks [D1], the medallion architecture descends from classic staging → ODS → mart patterns.

Layer	Contract	Consumer
Bronze	Raw, immutable, replayable, source-faithful	Pipelines only
Silver	Cleaned, typed, deduplicated, conformed entities	Engineers, downstream models

Layer	Contract	Consumer
Gold	Business-level, consumption-optimized	BI, ML features, apps

1.2 The agent reinterpretation

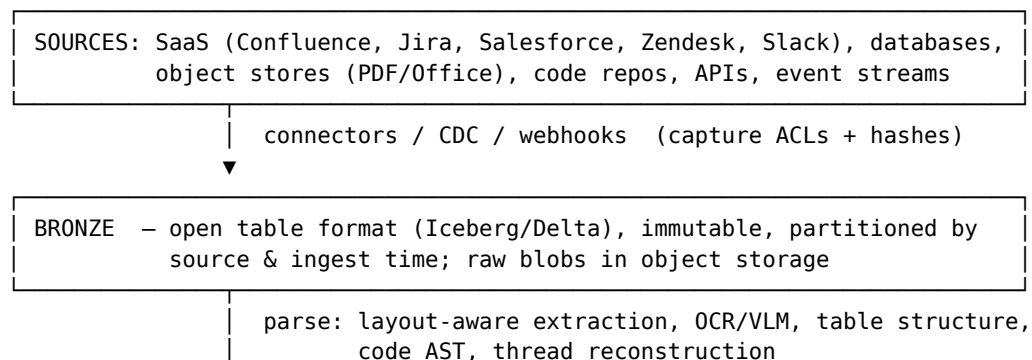
Layer	Agent-oriented contract
Bronze	Raw bytes + source metadata + source ACLs captured at ingest + content hash. <i>Never retrieved by an agent.</i>
Silver	Parsed, structured, linked: documents as structured markdown/JSON with layout preserved, entities resolved to canonical IDs, temporal validity attached, PII classified.
Gold	Task-aligned knowledge products: canonical, versioned, attributable, conflict-resolved artifacts with retrieval-optimized representations and a measured quality SLA.

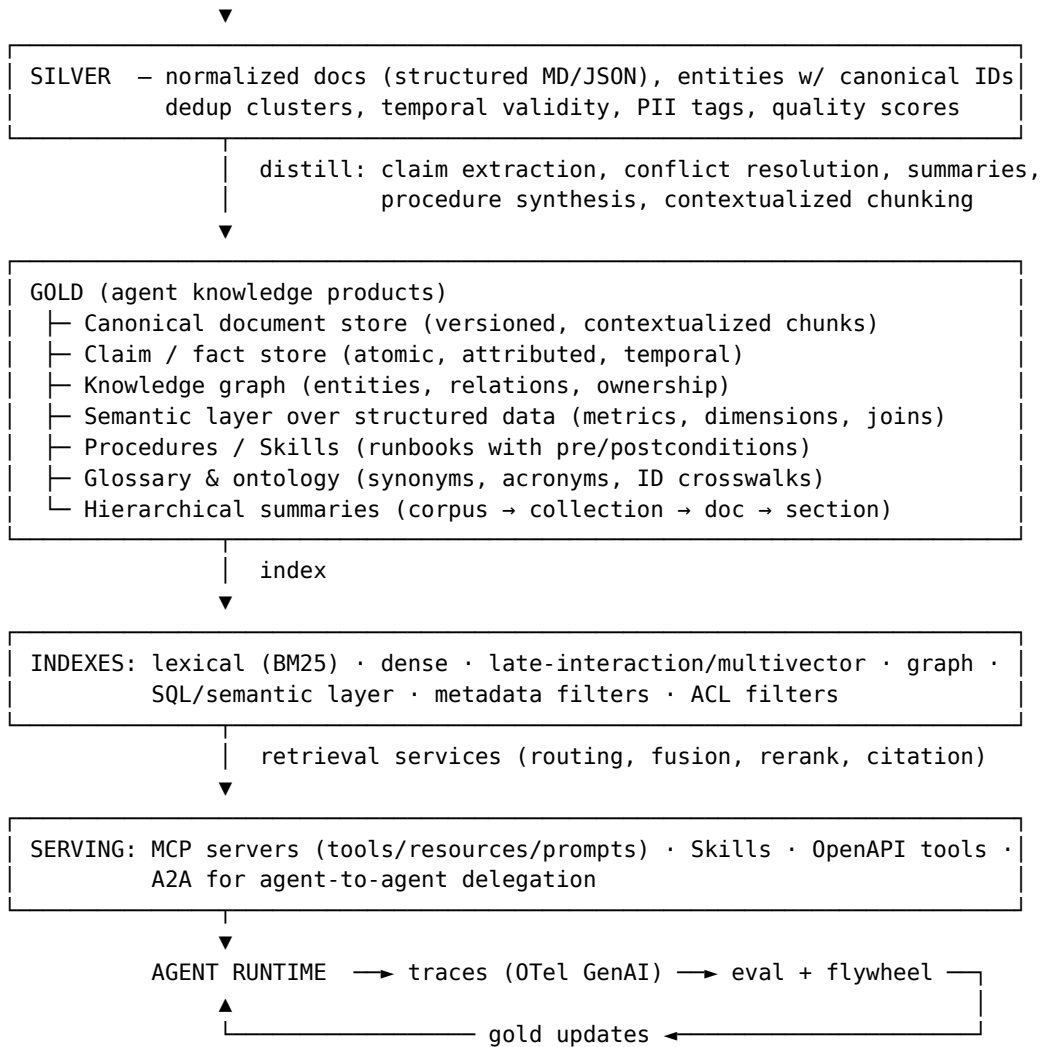
1.3 Properties of agent-grade gold

A gold artifact must satisfy all of the following:

Property	Definition	Failure if absent
Canonical	One authoritative answer per fact, or an explicit, flagged conflict	Agent picks arbitrarily between contradictions
Attributable	Every claim traces to a source span (doc, version, offset)	Unverifiable answers; no citation
Temporal	valid_from / valid_to / observed_at	Stale answers delivered confidently
Scoped	Applicability (region, product, tenant, role)	Correct fact, wrong context
Permissioned	ACLs inherited from source	Data exfiltration through the agent
Self-contained	Retrievable units are interpretable in isolation	Chunks that say “as noted above...”
Actionable	Procedures with preconditions, steps, failure modes	Agent knows <i>what</i> but not <i>how</i>
Measured	Covered by eval cases with known-correct answers	Silent regressions

2. Reference Architecture





Cross-cutting planes: lineage (OpenLineage), data contracts (ODCS), catalog and governance (Unity Catalog / Polaris / DataHub), security (ACL propagation, injection scanning), and evaluation (CI-gated).

3. Phase 1: Requirements and the Evaluation-First Contract

Work backward from the agent’s job. OpenAI’s agent-building guide and Anthropic’s “Building Effective Agents” both stress starting from concrete use cases and adding complexity only when measured need justifies it [O1][A1].

3.1 Task inventory

For each agent capability, specify:

- **Intent:** “Diagnose a failed payment for a customer.”
- **Required knowledge:** policy docs, payment state machine, error code catalog, customer records.
- **Required actions:** read ledger, open ticket, issue refund (with approval).
- **Answer form:** a cited diagnosis plus a recommended action.
- **Risk tier:** read-only, reversible write, or irreversible write.

3.2 Query taxonomy

Classify expected queries. Each class implies a different gold representation and retrieval path:

Query class	Example	Best representation
Exact lookup	“What does error E4021 mean?”	Lexical index, glossary, structured table
Semantic lookup	“How do we handle chargebacks in the EU?”	Contextualized chunks, hybrid retrieval
Aggregation / analytics	“Refund rate by region last quarter”	Semantic layer → SQL (never RAG over CSVs)
Multi-hop relational	“Who owns the service that emits E4021?”	Knowledge graph
Global / sensemaking	“What are the recurring themes in Q3 incidents?”	Hierarchical summaries, GraphRAG communities [R7]
Procedural	“Rotate the signing key”	Procedure / Skill artifact
Temporal	“What was the policy as of March?”	Bitemporal claim store
Live state	“Is the payment API degraded right now?”	Tool call to a live system (not gold)

Key boundary

Gold is for *curated knowledge*. Live operational state belongs behind tools that query systems of record directly. Do not snapshot fast-changing state into a vector index.

3.3 Build the golden evaluation set first

Before writing pipeline code, assemble 100–500 representative questions. Each case should include:

- the question, sourced from real tickets, search logs, and SME interviews (not invented by engineers);
- the expected answer;
- **expected source spans** (document ID + section);
- the query class and difficulty;
- negative cases: unanswerable questions where the correct behavior is “I don’t know,” and permission-denied cases.

This set becomes your retrieval benchmark, regression suite, and acceptance contract (Section 11).

3.4 Data contract for the gold layer

Define the gold layer’s SLA explicitly, ideally as an Open Data Contract Standard (ODCS) document [S3]:

```
dataProduct: payments-knowledge-gold
version: 2.3.0
owner: payments-platform@corp
sla:
  freshness: { policy_docs: "1h", error_catalog: "15m", tickets: "24h" }
  retrieval_recall_at_10: ">= 0.90"          # on golden set
  citation_faithfulness: ">= 0.95"
  unresolved_conflict_rate: "<= 1%"
```

```
schema: ./schemas/gold_chunk.json
access: inherit_source_acl
```

4. Phase 2: Source Inventory and Ingestion (Bronze)

4.1 Source inventory

For each source, record:

- system of record vs. derivative (e.g., a Confluence copy of a policy vs. the policy repo);
- **authority rank** (used later for conflict resolution);
- change mechanism (CDC, webhooks, polling, full dump);
- permission model (how ACLs are expressed and how to extract them);
- volatility (expected update frequency);
- content types (prose, tables, scans, code, threads, images);
- legal constraints (retention, residency, PII class).

4.2 Ingestion patterns

Pattern	Use when	Technology
CDC	Relational sources	Debezium, Fivetran/Airbyte CDC, native log replication
Webhooks + reconcile	SaaS with events	Event bus (Kafka/Kinesis) + periodic full reconcile
Incremental polling	APIs with updated_since	Airbyte, custom connectors
Full snapshot	Small or legacy sources	Scheduled batch

Always run a periodic full reconcile

Even with CDC or webhooks. Event streams drop deletes, and missed deletes are how revoked documents stay retrievable.

4.3 Bronze contract

- **Immutable and append-only.** Store raw bytes in object storage and metadata in an open table format (Apache Iceberg or Delta Lake) for time travel and replay.
- **Capture with every record:** source_system, source_id, source_uri, source_version, ingested_at, content_sha256, acl_snapshot, source_modified_at, and a deleted tombstone.
- **Capture ACLs at ingest.** Reconstructing permissions later is often impossible.
- **Lineage:** emit OpenLineage events per job [S4].

Rule: bronze is replayable. Every downstream layer must be rebuildable from bronze plus code. This matters every time you change a parser, chunker, or embedding model, which will be often.

5. Phase 3: Parsing and Normalization

Parsing is the most underestimated stage. Garbage extraction caps every downstream metric, and no reranker recovers a table that was flattened into word salad.

5.1 Document parsing

Content	Approach	Tools
Born-digital PDF/Office	Layout-aware extraction preserving headings, lists, reading order	Docling [P1], Unstructured, Marker, NVIDIA NeMo Retriever extraction (formerly NV-Ingest) [N2]
Scanned / complex layout	OCR + layout models, or VLM-based parsing	Docling, VLM pipelines, Azure Document Intelligence, Textract
Tables	Table-structure recognition; emit as HTML/markdown plus a structured row store	Docling TableFormer, NeMo Retriever table extraction
Charts / figures	VLM captioning + keep the image for multimodal retrieval	VLMS; ColPali-style page retrieval [R10]
Slides	Per-slide units with speaker notes	Same toolchain

Output format: structured markdown or a document JSON with a hierarchy (title → section → subsection → block) and stable block IDs. The heading path is critical metadata for chunk contextualization later.

Visual-native alternative: ColPali embeds page images directly with a VLM and late-interaction scoring [R10]. It skips OCR for visually rich documents and performs strongly on the ViDoRe benchmark. Use it as a parallel index for scan-heavy or diagram-heavy corpora, not as a replacement for text extraction where text extraction works.

5.2 Non-document sources

- **Conversations (Slack, Teams, email):** reconstruct threads; resolve users to canonical people IDs; strip signatures and quoted replies; mark *status* (decision vs. speculation). Conversations are low-authority by default.
- **Tickets:** separate problem, diagnosis, and resolution fields. Resolved tickets are raw material for procedures and FAQ distillation.
- **Code:** parse with ASTs (tree-sitter); chunk by symbol (function/class); attach file path, repo, commit, and call-graph edges.
- **Structured data:** do not embed rows. Model it in a semantic layer (Section 7.5) and expose it via text-to-SQL or metric tools.

5.3 Normalization

- Unicode NFKC, whitespace and boilerplate removal (nav bars, footers, cookie banners).
- Language detection and, if needed, translation with the original retained.
- Date normalization to ISO 8601 with timezone.
- Unit normalization where domain-relevant.

6. Phase 4: Conformance (Silver)

6.1 Deduplication

Duplicates poison retrieval. Top-k fills with near-copies, diversity collapses, and old versions outrank new ones. Deduplication measurably improves LM training [R14], and the same logic applies to retrieval corpora.

Level	Method
Exact	Content hash (SHA-256 on normalized text)
Near-duplicate	MinHash + LSH [R15]; SimHash
Semantic	Embedding-similarity clustering (e.g., NeMo Curator semantic dedup [N1])

Do not just delete duplicates. Cluster them, select a canonical representative (by authority rank, then recency), and keep the others as provenance links. “Five sources agree” is a useful confidence signal.

6.2 Entity resolution

Agents fail at joins humans do implicitly (“Acme,” “ACME Corp,” and cust_00481 are the same customer).

- Build a **canonical entity registry** with stable IDs for customers, products, services, people, and locations.
- Use probabilistic record linkage (Fellegi–Sunter model [R16]; Splink is a production-grade open implementation).
- Use LLM-assisted matching for ambiguous pairs, with human review queues for low-confidence merges.
- Maintain **ID crosswalks**: CRM ID ↔ billing ID ↔ support ID.
- Run entity linking (mention → canonical ID) over all silver text so chunks carry entities: [...] metadata.

6.3 Temporal modeling

Use a **bitemporal** model for facts:

- `valid_time`: when the fact is true in the world (policy effective dates);
- `transaction_time`: when your system learned it.

This answers both “What is the policy?” and “What did we believe the policy was on March 3?” The second question matters in audits and incident reviews.

6.4 Quality scoring and filtering

Assign per-document quality signals:

- extraction confidence (OCR/parse quality);
- boilerplate ratio, length, language;
- staleness (age relative to source volatility);
- authority (source rank);
- LLM-based quality classifiers (NeMo Curator provides classifier-based filtering patterns [N1]).

Quarantine low-quality content rather than silently dropping it.

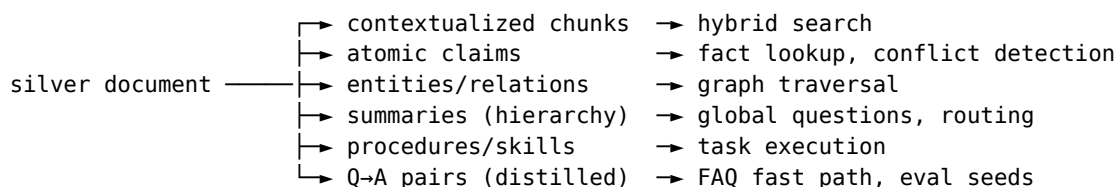
6.5 PII and sensitivity classification

- Detect PII and secrets (Microsoft Presidio, cloud DLP, secret scanners).
- Tag at the block level. Redact, tokenize, or restrict according to policy.
- **Scrub secrets before embedding.** Embeddings are partially invertible; research has demonstrated substantial text reconstruction from embeddings [R17]. Treat vector stores as containing the source data.

7. Phase 5: Gold Artifact Design

This is where agent gold diverges completely from warehouse gold.

7.1 Principle: multiple representations of one truth



All representations share provenance back to the same silver block IDs. This makes them consistent, jointly updatable, and jointly deletable.

7.2 Canonical document store and chunking

Chunking is a retrieval-unit design decision, not a string-splitting utility.

Strategy	Notes
Fixed-size token windows	Baseline. Cheap, ignores structure.
Structure-aware (heading/section)	Strong default for well-structured documents.
Page-level	NVIDIA's chunking evaluation found page-level chunking had the best average accuracy across their tested datasets; results varied by dataset and query type [N4].
Semantic chunking	Splits on embedding-similarity breakpoints. Mixed evidence; measure before adopting.
Propositions	Atomic, self-contained statements. Chen et al. showed proposition-level retrieval improved downstream QA over passage/sentence units in their benchmarks [R13].
Late chunking	Embed the full document with a long-context model, then pool per chunk, so chunk vectors retain document context [R18].
Parent-child (small-to-big)	Retrieve small precise units; return the enclosing section to the LLM.

Contextual retrieval (Anthropic, 2024) [A2]: prepend an LLM-generated, document-aware context sentence to each chunk before embedding *and* before BM25 indexing. In Anthropic's experiments, the top-20 retrieval failure rate fell by:

- 35% with contextual embeddings alone,
- 49% with contextual embeddings + contextual BM25,
- 67% when reranking was added.

Prompt caching makes this economically feasible at corpus scale.

Recommended default

Structure-aware chunks (~200–800 tokens, tuned on your eval set), plus a deterministic header (title › section path › effective date) and an LLM-generated context line, with parent-section expansion at serving time.

7.3 Gold chunk schema (example)

```

{
  "chunk_id": "pol-refunds-eu@v7#s3.2#c1",
  "doc_id": "pol-refunds-eu",

```

```

"doc_version": 7,
"text": "Refunds for EU consumers must be issued within 14 days of ...",
"context_header": "Refund Policy (EU) > 3.2 Consumer withdrawals. Effective
2026-01-01.",
"generated_context": "Statutory refund deadlines for EU consumer withdrawals.",
"heading_path": ["Refund Policy (EU)", "Timelines", "Consumer withdrawals"],
"content_type": "policy",
"entities": ["region:EU", "process:refund", "regulation:consumer-rights-directive"],
"authority": 0.95,
"valid_from": "2026-01-01",
"valid_to": null,
"observed_at": "2026-09-30T12:04:11Z",
"scope": { "region": ["EU"], "product": ["*"], "tenant": "*" },
"acl": { "allow_groups": ["support", "finance"], "deny_groups": [] },
"source": {
  "system": "confluence",
  "uri": "https://wiki.corp/x/AbC123",
  "version": "412",
  "span": { "block_ids": ["b_118", "b_119"] }
},
"supersedes": ["pol-refunds-eu@v6#s3.2#c1"],
"conflicts": [],
"quality": { "parse_confidence": 0.99, "staleness_days": 3 },
"embedding": { "model": "embed-model-x@2026-06", "dim": 1024 }
}

```

Note the embedding model version field. You will re-embed, and mixing vector spaces silently destroys recall.

7.4 Claim store and conflict resolution

Extract **atomic claims** (subject–predicate–object + qualifiers) with span-level provenance:

```

(claim) refund_window(region=EU, customer=consumer) = 14 days
  source: pol-refunds-eu@v7 §3.2 | authority 0.95 | valid_from 2026-01-01
(claim) refund_window(region=EU) = 30 days
  source: slack#support 2025-11-02 | authority 0.2

```

Conflict resolution policy. Decide this deliberately rather than leaving it to the LLM:

1. Same scope, different values → conflict detected.
2. Resolve by authority rank → validity period → recency → corroboration count.
3. If unresolvable: mark conflicts: [...], surface both with sources, and route to the owner's review queue.
4. Expose conflict status to the agent so it can hedge or escalate instead of guessing.

The claim store also supports **coverage analysis**: which entities or topics lack authoritative claims.

7.5 Structured data: semantic layer, not embeddings

For quantitative questions, the gold artifact is a **semantic layer**: governed metrics, dimensions, join paths, and business definitions. Examples include dbt Semantic Layer/MetricFlow, Cube, LookML, and Snowflake/Databricks metric views.

Why: text-to-SQL over raw schemas remains unreliable on realistic enterprise databases. Benchmarks such as BIRD [R19] and Spider 2.0 [R20] show large gaps between model and human performance on messy,

real-world schemas. A semantic layer converts “write correct SQL over 400 tables” into “pick the metric and filters,” which is a much easier problem.

Expose it via tools like `query_metric(metric, dimensions, filters, time_range)` rather than raw `execute_sql`, with SQL fallback for power use under read-only, row-level-secured credentials.

7.6 Knowledge graph

Use a graph when queries are multi-hop or relational (ownership, dependencies, org structure, lineage).

- **Property graph** (Neo4j, Neptune, etc.; ISO GQL standardized in 2024 as ISO/IEC 39075) or **RDF** with SHACL validation for ontology-heavy domains.
- Populate from structured sources first (CMDB, HRIS, service catalogs), since these are high precision. Then add LLM-extracted relations from text, with confidence scores.
- **GraphRAG** (Microsoft Research) [R7]: extract an entity graph, detect communities (Leiden), and pre-generate community summaries. It answers global “sensemaking” questions that chunk retrieval cannot. It is expensive to build, so justify it with eval data.
- Ontology discipline: a small, explicit schema of entity and relation types beats an unbounded LLM-invented graph that becomes a hairball.

7.7 Hierarchical summaries

Use RAPTOR-style recursive summarization [R8]: cluster chunks, summarize clusters, and repeat to build a tree. Index all levels. This serves:

- broad questions (retrieve high-level nodes);
- routing (decide which collection to search);
- “table of contents” resources the agent can browse via progressive disclosure.

7.8 Procedures and Skills

Procedural knowledge is the most valuable and most neglected gold artifact. Synthesize it from runbooks, resolved tickets, and SME interviews into a structured form:

```

procedure: rotate-payment-signing-key
version: 4
owner: payments-sre
preconditions:
  - role: payments-sre
  - no active incident tagged payments-sev1
inputs: [environment]
steps:
  - id: 1
    action: tool:kms.create_key_version
    args: { key: "payments-signing", env: "{environment}" }
    verify: "new version status == ENABLED"
  - id: 2
    action: tool:deploy.rollout
    requires_approval: true
failure_modes:
  - symptom: "signature mismatch errors spike"
    remedy: "roll back via step 2 with previous version"
sources: [runbook-114@v9, INC-2291 postmortem]
```

Anthropic Agent Skills (Oct 2025) [A5] package procedures as folders (SKILL.md + scripts + resources) loaded via **progressive disclosure**: only a short description sits in context until the skill is relevant. This is an excellent delivery format for procedural gold.

7.9 Glossary and ontology

Include acronyms, synonyms, internal codenames, deprecated names, and ID formats. Use them for:

- query expansion (lexical recall on jargon);
- entity linking;
- injection into the system prompt or a resource for domain grounding.

7.10 Distilled Q→A pairs

Generate candidate Q→A pairs from gold documents (Promptagator/InPars-style synthetic query generation [R21]), then have SMEs validate the high-traffic ones. Uses:

- FAQ fast path (high-precision exact matches);
- eval set expansion;
- training data for fine-tuning embedders and rerankers on your domain.

8. Phase 6: Indexing and Retrieval

8.1 Hybrid retrieval is the baseline

Dense retrieval misses exact identifiers, error codes, and rare jargon. Lexical retrieval misses paraphrase. BEIR showed BM25 remains a strong zero-shot baseline that many dense models failed to beat out of domain [R11]. Production systems combine both.

Reciprocal Rank Fusion (RRF) [R12]:

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + \text{rank}_r(d)}, \quad k \approx 60 \quad (1)$$

RRF is simple, robust, and requires no score calibration across retrievers.

8.2 Retrieval stack

query

- rewrite / decompose (multi-hop) / expand (glossary, HyDE [R22])
- route (query class → retriever(s)) ← Section 3.2 taxonomy
- candidate generation (BM25 u dense u graph u metadata filter), k≈50–200
- ACL + scope + validity filter (pre-filter, enforced in the index)
- fusion (RRF)
- rerank (cross-encoder or LLM reranker), keep k≈5–20
- dedup / diversify (MMR)
- expand (parent section, linked entities)
- package with citations + metadata

8.3 Components

Component	Options	Notes
Embedding model	Choose via MTEB [R23] and your own eval set	Public leaderboards are not your distribution. NVIDIA NV-Embed [N5] and

Component	Options	Notes
		other leading models; fine-tune on domain Q→A pairs for meaningful gains.
Late interaction	ColBERTv2 [R9], ColPali [R10]	Token-level matching; strong on precise queries; larger index.
Vector index	HNSW [R24], DiskANN [R25], IVF-PQ	pgvector, OpenSearch, Elasticsearch, Vespa, Milvus, Qdrant, Weaviate, LanceDB, Turbopuffer, cloud-native.
Compression	Matryoshka embeddings [R26], scalar/binary quantization	Cut cost substantially; validate recall impact on your eval set.
Reranker	Cross-encoders (e.g., NVIDIA NeMo Retriever rerankers [N2], Cohere, BGE), LLM-as-reranker	Typically the highest-ROI single addition after hybrid search.
Lexical	BM25 (OpenSearch/Elasticsearch/Tan-tivy), SPLADE (learned sparse)	Index the contextualized text, not raw chunks.

Engine selection heuristic

Choose an engine that does **hybrid search + metadata filtering + ACL filtering natively in one query**. Stitching together separate vector and keyword stores and filtering ACLs after retrieval is the most common source of both recall loss and leakage.

8.4 Adaptive and agentic retrieval

- **Self-RAG** [R5]: the model decides when to retrieve and critiques relevance and support.
- **Corrective RAG (CRAG)** [R6]: evaluates retrieval quality and triggers fallback or re-query when confidence is low.
- **Agentic search**: the agent issues multiple targeted queries via tools, inspects results, and refines. Anthropic notes the shift toward “just-in-time” retrieval, where agents hold lightweight references (paths, IDs, queries) and load data on demand [A3].

Implication for gold design: expose retrieval as *composable tools* (search, get-document, get-entity, list-related, query-metric) rather than one opaque “ask the knowledge base” endpoint. Agents increasingly drive retrieval themselves.

8.5 Long context does not replace retrieval

- Liu et al., “Lost in the Middle” [R3]: accuracy is highest when relevant information is at the start or end of the context and degrades for information in the middle.
- NVIDIA’s RULER [R4]: many models fail to sustain performance up to their claimed context lengths on tasks harder than needle retrieval.
- Chroma’s “Context Rot” study (2025) [R27]: performance degrades as input length grows, even on simple tasks.

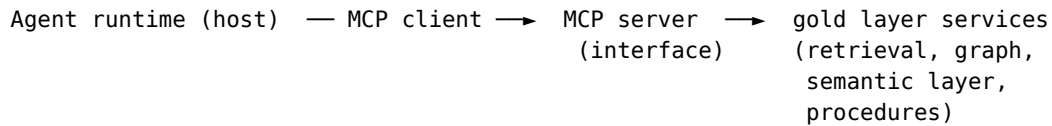
Long context raises the ceiling on *how much* relevant material you can include. It does not remove the need for precision. Retrieval quality still dominates, and token cost and latency scale with stuffing.

9. Phase 7: Serving Interfaces: MCP, Tools, Skills, A2A

9.1 Where MCP fits

Model Context Protocol (MCP) is an open protocol, introduced by Anthropic in November 2024, that standardizes how AI applications connect to external data and tools [M1]. It has been adopted by OpenAI (Agents SDK and Responses API, 2025), Google, Microsoft, and most major agent frameworks. In December 2025 it was contributed to the Linux Foundation’s Agentic AI Foundation.

Position in the stack:



MCP is the **interface contract** between agents and your gold layer. It solves the N×M integration problem: write one server, and any MCP-capable agent can use it. It is **not** a database, retrieval algorithm, or permission system by itself.

9.2 MCP primitives mapped to gold artifacts

MCP primitive	Control	Gold mapping
Tools	Model-invoked	search_knowledge, get_document, get_entity, traverse_graph, query_metric, run_procedure_step
Resources	Application-selected, URI-addressed	gold://docs/{id}@{version}, gold://glossary, gold://summaries/{collection}, entity pages
Prompts	User-selected templates	“Investigate incident,” “Draft customer response with citations”
Client features: sampling, roots, elicitation	Server → client	Elicitation for confirming risky actions or missing parameters

Key spec evolution (verify the current revision at modelcontextprotocol.io):

- **2025-03-26:** Streamable HTTP transport replaced HTTP+SSE; OAuth 2.1 authorization framework.
- **2025-06-18:** structured tool output (outputSchema), elicitation, and MCP servers classified as OAuth Resource Servers with RFC 8707 resource indicators to prevent token misuse.

- **Later revisions:** continued work on async operations, auth, and registry/discovery. Pin your SDK and track the changelog.

9.3 Designing an MCP server for gold data

Anthropic’s “Writing effective tools for agents” [A4] and OpenAI’s agent guide [O1] converge on the following principles:

1. **Design for tasks, not endpoints.** One `find_customer_context(customer)` that returns a consolidated, cited view beats five CRUD wrappers the agent must orchestrate.
2. **Return high-signal, token-efficient output.** Support `response_format: "concise" | "detailed"`, pagination, and truncation with guidance (“37 more results; refine by date”).
3. **Return semantic identifiers and citations.** Use human-readable names alongside IDs, plus `source_uri` and version on every item.
4. **Write tool descriptions as if onboarding a new engineer.** Include when to use the tool, when not to, parameter semantics, and examples. Tool descriptions are prompts.
5. **Return actionable errors.** “No results for E4021; did you mean E4021?” beats 404.
6. **Enforce permissions server-side using the end user’s identity.** Pass user identity via OAuth and filter in the index. Never rely on the model to respect access rules.
7. **Make it evaluable.** Every tool should be exercised by the eval suite.

Example (Python, official MCP SDK’s FastMCP interface):

```
from mcp.server.fastmcp import FastMCP, Context
from pydantic import BaseModel

mcp = FastMCP("payments-knowledge")

class Hit(BaseModel):
    chunk_id: str
    title: str
    snippet: str
    source_uri: str
    doc_version: int
    valid_from: str | None
    authority: float
    conflicts: list[str]

class SearchResult(BaseModel):
    hits: list[Hit]
    total: int
    note: str | None = None

@mcp.tool()
async def search_knowledge(
    query: str,
    region: str | None = None,
    as_of: str | None = None,
    limit: int = 8,
    ctx: Context = None,
) -> SearchResult:
    """Search curated payments knowledge (policies, runbooks, error catalog).

    Use for: policy questions, error meanings, how-to procedures.
    Do NOT use for: live system status (use `payment_api_status`) or metrics
```

```

    (use `query_metric`). Results include citations; always cite `source_uri`.
    `as_of` (ISO date) returns knowledge valid at that date.
    """
    user = ctx.request_context.user          # identity from OAuth
    hits, total = await retrieval.hybrid_search(
        query=query, user=user, region=region, as_of=as_of, k=limit
    )
    # ACL + validity filtered in-index
    note = None if total <= limit else f"{total - limit} more; narrow with region/as_of."
    return SearchResult(hits=hits, total=total, note=note)

@mcp.resource("gold://docs/{doc_id}")
async def get_doc(doc_id: str) -> str:
    """Full canonical document (latest version) as markdown with section anchors."""
    return await store.render_markdown(doc_id)

```

9.4 Adjacent standards and patterns

Standard / pattern	Role	Relation to gold
OpenAPI	Describe HTTP APIs	Many frameworks auto-convert OpenAPI → tools; still curate descriptions.
OpenAI function calling / Structured Outputs [O2]	Schema-constrained tool args and outputs	Guarantees valid JSON against your schemas.
Agent Skills [A5]	Packaged procedures + scripts with progressive disclosure	Delivery format for procedural gold (Section 7.8).
A2A (Agent2Agent) [M2]	Agent-to-agent delegation protocol (Google, Apr 2025; now Linux Foundation)	Use when a “payments expert agent” owns the gold domain and other agents delegate to it. MCP = agent ↔ tools/data; A2A = agent ↔ agent.
AGENTS.md	Repo-level instructions for coding agents	Gold for code-focused agents.
llms.txt	Proposed convention for LLM-friendly site/docs indexes	Cheap win for public documentation corpora.
OpenTelemetry GenAI semantic conventions	Standard tracing for LLM/agent calls	Required for evaluation and the flywheel (Section 14).

9.5 Single agent vs. domain-expert agent

- **Single agent + MCP gold server:** simplest, and the right default [A1].

- **Domain agent behind A2A or as a sub-agent:** use when the domain needs its own reasoning, memory, policies, or a different model. Anthropic reports that multi-agent research systems use substantially more tokens than single agents [A6], so the gain must justify the cost.

10. Context Engineering at Inference Time

The gold layer determines what *can* reach the model. Context engineering determines what *does* [A3].

10.1 Context budget allocation

System prompt: role, domain rules, glossary excerpt	stable → prompt-cacheable
Tool definitions (MCP tool list)	stable → cacheable; keep lean
Skills: descriptions only (progressive disclosure)	
Task state / plan / memory notes	
Retrieved evidence (ranked, cited, deduped)	most relevant at start/end
Conversation (compacted)	

10.2 Practices

- **Fewer, better tokens.** Rerank aggressively; send 5–15 strong chunks, not 50 mediocre ones.
- **Position matters.** Put the strongest evidence first or last, not buried in the middle [R3].
- **Format evidence consistently.** Use XML/JSON blocks with source, version, valid_from, authority, and conflicts. Instruct the model to cite by ID and to say “not found” when evidence is absent.
- **Prune the tool surface.** Large tool lists degrade selection accuracy and consume context. Load tool subsets dynamically by task, or use tool-search patterns.
- **Compaction and external memory.** Summarize long trajectories; persist structured notes outside the context window [A3]. MemGPT [R28] formalized tiered memory for LLMs.
- **Prompt caching.** Stable prefixes (system prompt, tools, glossary) should be cache-friendly. Order content from most stable to least stable.

11. Evaluation

Evaluate each layer separately, then end to end. If you only measure final answers, you cannot tell whether retrieval or generation failed.

11.1 Metric stack

Layer	Metric	How
Parsing	Extraction accuracy, table cell accuracy	Sampled human review against source
Silver	Dedup precision/recall, entity-resolution precision/recall	Labeled pairs
Gold	Conflict rate, coverage (eval questions with an authoritative source), freshness lag	Automated checks
Retrieval	Recall@k, MRR, nDCG against expected source spans	Golden set

Layer	Metric	How
Generation	Faithfulness / groundedness (claims supported by retrieved context), answer correctness, citation precision/recall [R29]	LLM judge + human calibration
Abstention	Correct “don’t know” rate on unanswerable cases; false-refusal rate	Negative cases
Security	Permission-leak rate (must be 0), injection success rate	Red-team suite
Agent / task	Task success, tool-call correctness, steps/cost/latency, pass ^k consistency	Trajectory evals (τ -bench style [R30])

11.2 Frameworks and methods

- **RAGAS** [R31]: reference-free RAG metrics (faithfulness, answer relevance, context precision/recall).
- **ARES** [R32]: trained LLM judges with prediction-powered inference for confidence intervals using a small human-labeled set.
- **ALCE** [R29]: citation quality evaluation.
- **OpenAI Evals** [O3], plus platforms such as Braintrust, LangSmith, Arize Phoenix, and Langfuse.
- **LLM-as-judge** [R33]: scalable, but with known biases (position, verbosity, self-preference). **Calibrate the judge against human labels** and report agreement. An uncalibrated judge is a random number generator with good manners.

11.3 Evaluation as CI

- Every change to parsers, chunkers, embedding models, rerankers, prompts, tool descriptions, or the model version runs the retrieval and end-to-end suites.
- Gate merges on non-regression of key metrics, with confidence intervals. Small eval sets produce noisy deltas.
- Version the eval set. Add every production failure as a new case.

11.4 τ -bench and consistency

τ -bench [R30] introduced **pass^k**: the probability an agent succeeds on *all* k trials of the same task. Agents that look acceptable on pass@1 often show sharply lower pass^k. For production, consistency matters more than peak performance, so measure it.

12. Operations: Freshness, Versioning, Observability

12.1 Incremental processing

- Drive everything from content hashes: an unchanged hash means skip parse, chunk, and embed.
- Propagate changes through the DAG (source block $\rightarrow$ chunks $\rightarrow$ claims $\rightarrow$ graph edges $\rightarrow$ summaries).
- **Deletes and permission changes are first-class events.** They must propagate to every index within the SLA. Measure deletion lag.

12.2 Versioning

Artifact	Version strategy
Source documents	Source version + content hash
Pipeline code / config	Git SHA recorded on every gold row
Embedding model	Versioned index namespace; never mix vector spaces
Gold data product	Semver; contract changes are breaking
Eval set	Versioned; metrics reported against a specific version

12.3 Re-embedding and index migrations

Use **blue/green indexes**. Build the new index (new model or chunking) in parallel, run the eval suite against both, shift traffic, and keep rollback available. Open table formats (Iceberg/Delta time travel) make this tractable for the underlying data.

12.4 Orchestration and processing

- **Orchestration:** Dagster (asset-oriented, a good fit for lineage of gold artifacts), Airflow, Prefect.
- **Processing:** Spark/Databricks, Ray (GPU-heavy parsing/embedding), NVIDIA NeMo Curator (GPU-accelerated dedup/classification at scale [N1]).
- **Transformations:** dbt for the structured/semantic-layer side.

12.5 Observability

- Trace every agent run with OpenTelemetry GenAI semantic conventions: query, rewrites, retrieved IDs, scores, reranked set, tool calls, tokens, latency, and final answer.
- Dashboards: retrieval empty-rate, low-confidence rate, conflict-surfaced rate, freshness lag per source, cost per task, and user feedback.
- **Store retrieved chunk IDs with every answer.** This is required for debugging, audit, and the flywheel.

13. Security and Governance

13.1 Threat model

The OWASP Top 10 for LLM Applications (2025) [S1] lists prompt injection as LLM01 and adds vector/embedding weaknesses (LLM08). Greshake et al. [R34] demonstrated **indirect prompt injection**: malicious instructions embedded in retrieved content hijack the model. **Your gold layer is an attack surface.** Anything an attacker can write into a source (a ticket, a wiki page, an email) can reach the model.

The “lethal trifecta” (Simon Willison)

Private data access + exposure to untrusted content + an exfiltration channel in one agent means data can be stolen. Break at least one leg.

13.2 Controls

Control	Implementation
ACL propagation	Capture at ingest → carry on every gold row → enforce as a pre-filter in the index using the end user’s identity. Test with permission-denied eval cases.

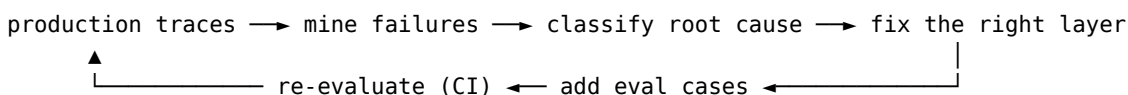
Control	Implementation
Authority-aware trust	Low-authority sources (chat, tickets, external web) are tagged and rendered to the model as <i>untrusted data</i> , never as instructions.
Injection scanning	Classifier-based detection at ingest (silver) and at retrieval; quarantine suspicious content. Defense in depth; no scanner is complete.
Least-privilege tools	Read-only by default; scoped OAuth tokens (MCP RFC 8707 resource indicators); human approval for irreversible actions.
Data poisoning defenses	Provenance on every claim; anomaly detection on sudden authoritative-claim changes; review queues for high-authority edits.
PII / secrets	Detect and redact before embedding; vectors inherit source sensitivity [R17].
Right to deletion	Deletion propagates to chunks, claims, graph, summaries, caches, <i>and</i> eval/ training sets.
MCP supply chain	Only allow vetted MCP servers; pin versions; review tool descriptions (tool-description poisoning is a documented attack vector).
Audit	Log who asked what, which data was retrieved, and which tools were invoked.

13.3 Governance frameworks

- **NIST AI RMF 1.0** [S2] and the **Generative AI Profile (NIST AI 600-1)** for risk-management structure.
- **ISO/IEC 42001** for AI management systems, if certification matters.
- **EU AI Act** obligations, depending on use case and jurisdiction.
- **Catalog and policy:** Unity Catalog (open-sourced 2024), Apache Polaris, DataHub, OpenMetadata.

14. The Data Flywheel: Closing the Loop

NVIDIA's Data Flywheel blueprint [N3] formalizes using production interaction data to continuously improve models and retrieval. The same loop applies to gold data:



Root-cause taxonomy (fix at the correct layer):

Symptom	Root cause	Fix
Answer missing / “don’t know”	Coverage gap	Author or ingest a source; SME request
Right doc exists, not retrieved	Retrieval	Chunking, contextualization, glossary, embedder fine-tune
Retrieved, but stale	Freshness	Ingestion SLA, validity metadata
Contradictory answer	Conflict resolution	Authority rules; owner review
Retrieved and correct, wrong answer	Generation	Prompt, evidence format, model
Wrong tool / bad args	Tool design	Descriptions, consolidation, schemas
Leaked data	ACL propagation	Stop the line; fix the filter; add regression

14.1 Serving data vs. training data

Keep these separate:

- **Serving gold:** facts, procedures, and tools retrieved at inference. It changes daily and must be attributable.
- **Training data:** trajectories, preference pairs, and hard negatives for fine-tuning embedders, rerankers, or the agent model.

Do not fine-tune facts into models when they belong in retrieval

Fine-tuned knowledge is unattributable, hard to update, and hard to delete. Fine-tune for *behavior* (format, tool use, domain reasoning style) and *retrieval components* (embedders, rerankers), where gains are well documented.

15. Technology Map

Function	Representative technologies
Ingestion / connectors	Airbyte, Fivetran, Debezium, Kafka/Kinesis, custom webhooks
Storage (bronze/silver)	S3/GCS/ADLS + Apache Iceberg or Delta Lake
Parsing	Docling, Unstructured, Marker, NVIDIA NeMo Retriever extraction, cloud document AI, VLMs
Curation / dedup	NVIDIA NeMo Curator, datasketch (MinHash), Spark
Entity resolution	Splink, Zingg, LLM-assisted matching
PII / secrets	Microsoft Presidio, cloud DLP, gitleaks/trufflehog
Orchestration	Dagster, Airflow, Prefect
Transformations / semantic layer	dbt (+ MetricFlow), Cube, LookML, warehouse metric views
Graph	Neo4j, Amazon Neptune, Kuzu, RDF stores; Microsoft GraphRAG
Search / vector	OpenSearch, Elasticsearch, Vespa, pgvector, Milvus, Qdrant, Weaviate, LanceDB, Turbopuffer
Embeddings / rerankers	NVIDIA NeMo Retriever (NIM), OpenAI, Cohere, Voyage, BGE/Jina/Nomic (open), ColBERT, ColPali
Serving interface	MCP SDKs (Python/TypeScript/etc.), OpenAPI, Agent Skills, A2A
Agent frameworks	OpenAI Agents SDK, Claude Agent SDK, LangGraph, LlamaIndex, Google ADK, Microsoft Agent Framework
Evaluation	RAGAS, ARES, OpenAI Evals, Braintrust, LangSmith, Arize Phoenix, Langfuse
Observability	OpenTelemetry GenAI conventions + any OTel backend
Governance / lineage	Unity Catalog, Apache Polaris, DataHub, OpenMetadata, OpenLineage, ODCS
End-to-end references	NVIDIA AI Blueprint for RAG, NVIDIA Data Flywheel Blueprint

This field moves quickly. Treat specific products as examples and select them by running your eval set, not by reading leaderboards or vendor benchmarks.

16. Anti-Patterns

1. **“Dump everything into a vector DB.”** Bronze served as gold. Noise, duplicates, secrets, and contradictions go straight to the model.
2. **Dense-only retrieval.** Fails on IDs, error codes, and jargon. Hybrid is the floor.
3. **Embedding tables and metrics.** RAG over CSV rows for “what was revenue in Q2” produces fabricated arithmetic. Use a semantic layer.
4. **Post-filtering ACLs.** Retrieving top-k and then dropping unauthorized results loses recall and risks leakage in logs and caches.
5. **No validity time.** The agent cheerfully quotes the 2023 policy.
6. **Letting the LLM resolve conflicts silently.** Conflict resolution is a data policy, not a sampling outcome.
7. **One giant ask_kb tool.** It is opaque, unevaluable, and prevents agentic retrieval.
8. **Endpoint-mirroring MCP servers.** Fifty CRUD tools that wrap a REST API one-to-one bloat context and confuse tool selection.
9. **Eval by vibes.** “It looked good in the demo” is not a metric.
10. **Mixed embedding versions in one index.** This causes silent recall collapse.
11. **Fine-tuning facts.** Facts belong in retrieval; fine-tuning is for behavior and retrieval components.
12. **Treating long context as a substitute for curation.** Context rot is measurable [R27].
13. **Trusting retrieved text as instructions.** This is the core of indirect prompt injection.

17. Maturity Model and Launch Checklist

17.1 Maturity levels

Level	Characteristics
L0: Demo	Files → chunk → embed → dense top-k. No eval. No ACLs.
L1: Baseline	Layout-aware parsing; hybrid + rerank; source citations; golden set with retrieval metrics.
L2: Production	Medallion pipeline with replay; dedup + entity resolution; ACL pre-filtering; validity metadata; contextualized chunks; MCP server with task-level tools; CI evals; OTel tracing.
L3: Advanced	Claim store with conflict resolution; knowledge graph; semantic layer; procedures/Skills; query routing by class; agentic retrieval; injection defenses; freshness SLAs.
L4: Self-improving	Data flywheel with root-cause triage; fine-tuned embedders/rerankers; coverage analytics driving SME authoring; pass ^k reliability targets.

17.2 Launch checklist

Data

- ☐ Every gold record has source URI, version, span, valid_from/valid_to, authority, scope, and ACL
- ☐ Bronze is immutable and the full pipeline replays from it
- ☐ Deletes and permission changes propagate end to end within the SLA (measured)
- ☐ Dedup clusters with canonical selection; entity IDs resolved
- ☐ Conflict policy implemented; unresolved conflicts surfaced and routed to owners
- ☐ PII and secrets handled before embedding

Retrieval

- ☐ Hybrid (lexical + dense) + rerank, with ACL/validity pre-filtering
- ☐ Query classes routed to suitable representations (SQL, graph, chunks, procedures)
- ☐ Single embedding model version per index; blue/green migrations

Serving

- ☐ MCP server with task-level tools, clear descriptions, structured outputs, pagination, citations, and actionable errors
- ☐ User identity flows via OAuth; authorization enforced server-side
- ☐ Irreversible actions require approval

Evaluation and operations

- ☐ Golden set with expected source spans, negative cases, and permission cases
- ☐ Retrieval, faithfulness, citation, abstention, and task-success metrics in CI
- ☐ LLM judge calibrated against human labels
- ☐ OTEL traces with retrieved IDs stored per answer
- ☐ Red-team suite for injection and leakage; leak rate = 0

Bottom line

Gold for agents is not a cleaner table. It is a governed, versioned, multi-representation knowledge product with provenance on every claim, permissions on every row, time on every fact, a task-shaped tool interface (MCP), and an eval suite that tells you when any of it breaks.

18. References

Verify current versions of specifications and documentation; this field revises quickly.

Industry: Anthropic

- [A1] Anthropic. “Building effective agents.” Dec 2024.
- [A2] Anthropic. “Introducing Contextual Retrieval.” Sep 2024.
- [A3] Anthropic. “Effective context engineering for AI agents.” 2025.
- [A4] Anthropic. “Writing effective tools for agents — with agents.” 2025.
- [A5] Anthropic. “Equipping agents for the real world with Agent Skills.” Oct 2025.
- [A6] Anthropic. “How we built our multi-agent research system.” Jun 2025.

Industry: OpenAI

- [O1] OpenAI. “A practical guide to building agents.” 2025.
- [O2] OpenAI. “Introducing Structured Outputs in the API.” Aug 2024; Function calling documentation.
- [O3] OpenAI. Evals framework and evaluation best-practices documentation.

Industry: NVIDIA

- [N1] NVIDIA. NeMo Curator documentation (deduplication, quality classification).
- [N2] NVIDIA. NeMo Retriever (extraction, embedding, reranking NIMs) and the AI Blueprint for RAG.
- [N3] NVIDIA. AI Blueprint: Data Flywheel. 2025.
- [N4] NVIDIA Technical Blog. “Finding the Best Chunking Strategy for Accurate AI Responses.” 2025.
- [N5] Lee et al. “NV-Embed: Improved Techniques for Training LLMs as Generalist Embedding Models.” arXiv:2405.17428, 2024.

Industry: Other

- [D1] Databricks. “What is a medallion architecture?” Documentation.
- [M1] Model Context Protocol specification and documentation. modelcontextprotocol.io.
- [M2] Agent2Agent (A2A) Protocol specification. a2a-protocol.org (Linux Foundation).

Standards and governance

- [S1] OWASP. “Top 10 for LLM Applications.” 2025 edition.
- [S2] NIST. AI Risk Management Framework 1.0 (2023); NIST AI 600-1 Generative AI Profile (2024).
- [S3] Bitol (Linux Foundation). Open Data Contract Standard (ODCS).
- [S4] OpenLineage specification (LF AI & Data).

Research

- [R1] Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.” NeurIPS 2020.
- [R2] Yao et al. “ReAct: Synergizing Reasoning and Acting in Language Models.” ICLR 2023.
- [R3] Liu et al. “Lost in the Middle: How Language Models Use Long Contexts.” TACL 2024.
- [R4] Hsieh et al. (NVIDIA). “RULER: What’s the Real Context Size of Your Long-Context Language Models?” COLM 2024.
- [R5] Asai et al. “Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection.” ICLR 2024.
- [R6] Yan et al. “Corrective Retrieval Augmented Generation.” arXiv:2401.15884, 2024.
- [R7] Edge et al. (Microsoft Research). “From Local to Global: A Graph RAG Approach to Query-Focused Summarization.” arXiv:2404.16130, 2024.
- [R8] Sarthi et al. “RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval.” ICLR 2024.
- [R9] Santhanam et al. “ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction.” NAACL 2022.
- [R10] Faysse et al. “ColPali: Efficient Document Retrieval with Vision Language Models.” ICLR 2025.
- [R11] Thakur et al. “BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models.” NeurIPS Datasets & Benchmarks 2021.
- [R12] Cormack, Clarke, Büttcher. “Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods.” SIGIR 2009.
- [R13] Chen et al. “Dense X Retrieval: What Retrieval Granularity Should We Use?” EMNLP 2024.
- [R14] Lee et al. “Deduplicating Training Data Makes Language Models Better.” ACL 2022.
- [R15] Broder. “On the Resemblance and Containment of Documents.” 1997.
- [R16] Fellegi & Sunter. “A Theory for Record Linkage.” JASA 1969.
- [R17] Morris et al. “Text Embeddings Reveal (Almost) As Much As Text.” EMNLP 2023.
- [R18] Günther et al. “Late Chunking: Contextual Chunk Embeddings Using Long-Context Embedding Models.” arXiv:2409.04701, 2024.
- [R19] Li et al. “Can LLM Already Serve as a Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs (BIRD).” NeurIPS 2023.
- [R20] Lei et al. “Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows.” ICLR 2025.
- [R21] Dai et al. “Promptagator: Few-shot Dense Retrieval From 8 Examples.” ICLR 2023.
- [R22] Gao et al. “Precise Zero-Shot Dense Retrieval without Relevance Labels (HyDE).” ACL 2023.
- [R23] Muennighoff et al. “MTEB: Massive Text Embedding Benchmark.” EACL 2023.
- [R24] Malkov & Yashunin. “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs.” IEEE TPAMI 2018.
- [R25] Subramanya et al. “DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node.” NeurIPS 2019.
- [R26] Kusupati et al. “Matryoshka Representation Learning.” NeurIPS 2022.
- [R27] Hong, Troynikov, Huber (Chroma). “Context Rot: How Increasing Input Tokens Impacts LLM Performance.” 2025.
- [R28] Packer et al. “MemGPT: Towards LLMs as Operating Systems.” arXiv:2310.08560, 2023.
- [R29] Gao et al. “Enabling Large Language Models to Generate Text with Citations (ALCE).” EMNLP 2023.
- [R30] Yao et al. (Sierra). “ τ -bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains.” arXiv:2406.12045, 2024.
- [R31] Es et al. “RAGAS: Automated Evaluation of Retrieval Augmented Generation.” EACL 2024 (demo).
- [R32] Saad-Falcon et al. “ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems.” NAACL 2024.
- [R33] Zheng et al. “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena.” NeurIPS 2023.

- [R34] Greshake et al. “Not What You’ve Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection.” AISC 2023.
- [P1] Auer et al. (IBM Research). “Docling Technical Report.” arXiv:2408.09869, 2024.