Semantics Delivery Network

Rethinking Web Retrieval Infrastructure for LLM Agents

Peichun Hua and Yunming Xiao

School of Data Science, The Chinese University of Hong Kong, Shenzhen

HotNets '26 · Salt Lake City



Paper: arXiv:2609.22486 · Code: github.com/sarendis56/SemDN_HotNets26

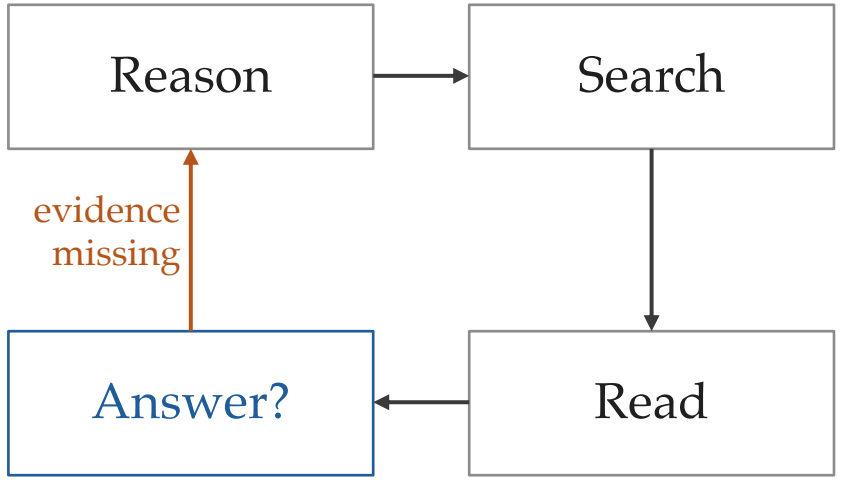
Outline

What this talk covers

1	**Today's retrieval path**	How agents read the web through infrastructure built for human readers
2	**Four mismatches**	Similarity vs utility, pages vs chunks, URLs vs semantics, repeated preprocessing
3	**Semantics Delivery Network**	One substrate that answers a query with chunks from the edge
4	**Research agenda**	Misses, reuse and freshness, shared state, incentives, and trust
5	**Open questions**	Where we would like your ideas

LLM agents now read the live web

- **Retrieval-augmented generation (RAG)** grounds an answer in passages fetched at query time
- **Agentic search** repeats the loop: reason, search, read, and search again
- **The live web** replaces static corpora for fresh and long-tail questions
- **Query-time fetcher traffic** from AI agents is rising sharply at CDNs

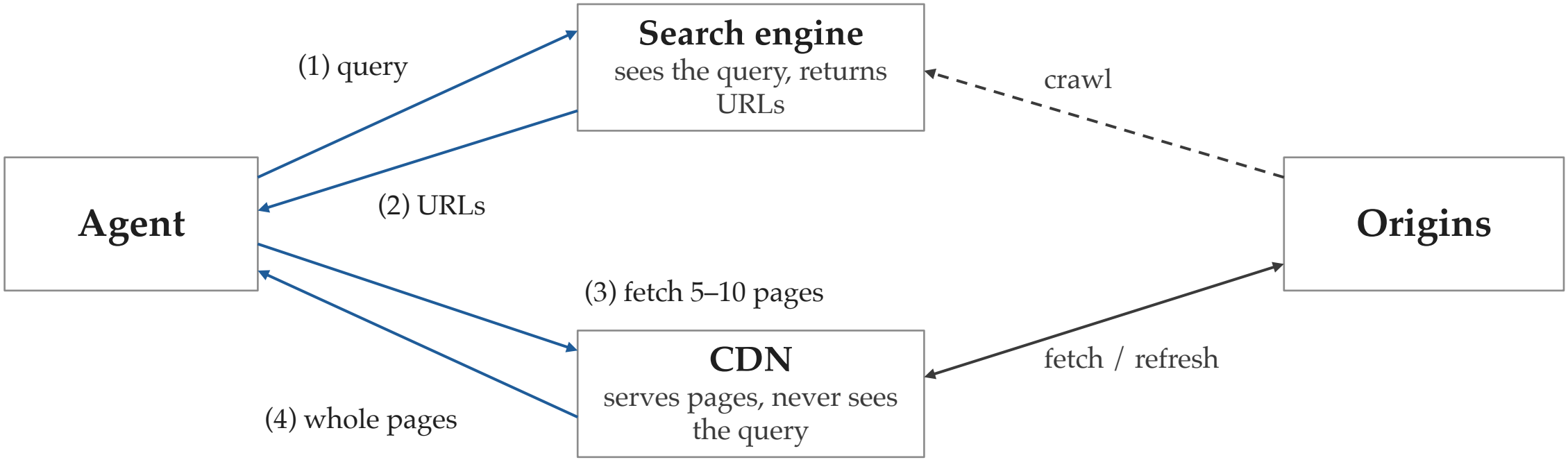


One question becomes a sequence of related sub-queries within a task.

Lewis et al., "Retrieval-Augmented Generation", NeurIPS 2020 · Jin et al., "Search-R1", arXiv 2025 · Fastly, "Q2 2025 Threat Insights Report", 2025

Today the job is split across two systems

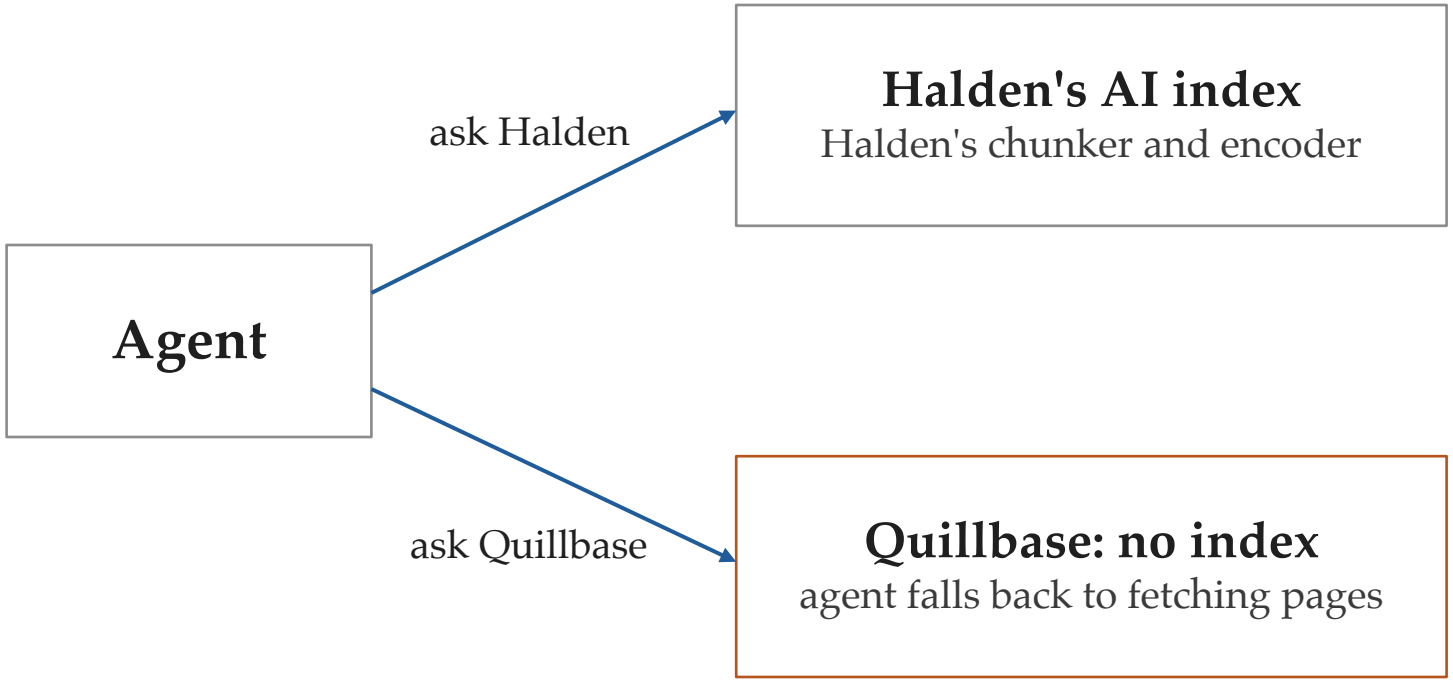
Request: *"Compare the API rate limits of Halden and Quillbase"*



(5) clean, chunk, embed, rank
done again by every operator; evidence missing → rewrite and repeat

The search engine decides what to read but returns only URLs. The CDN delivers from the edge but cannot see what the agent needs.

Per-site AI indexes help one site at a time

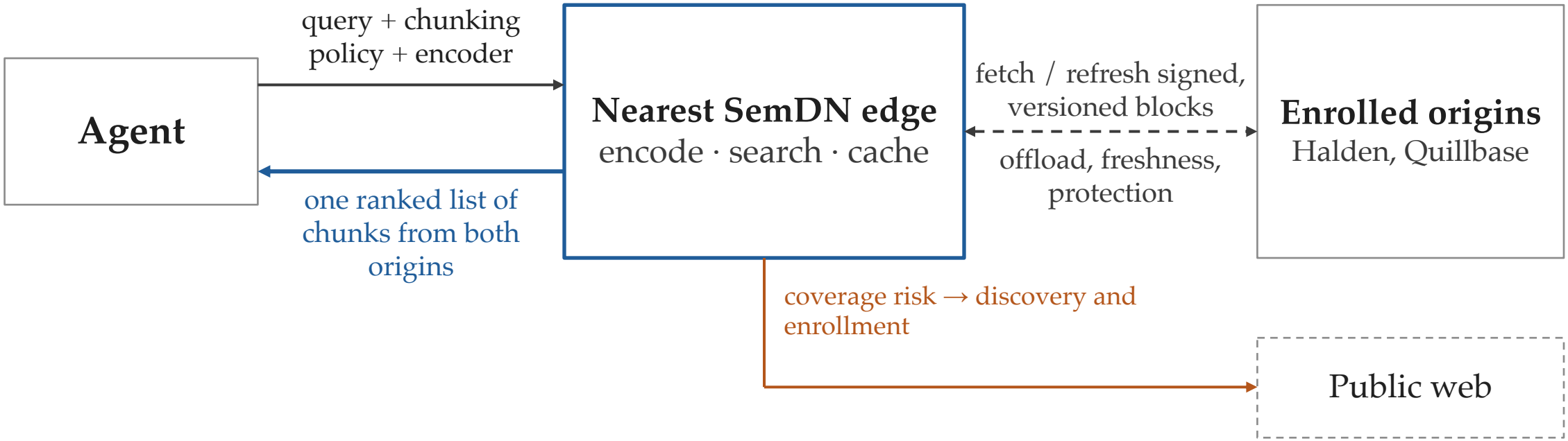


- Two rankings in two representations cannot be compared
- The agent re-embeds both lists and ranks them itself
- **AISTRAL** adds a resolver that picks which sites to ask; ranking across sites stays with the agent

Both are organized around the origin: one index and one representation per site.

Cloudflare, "An AI Index for All Our Customers", 2025 · Cloudflare, "AI Search", 2026 · Chen et al., "From Crawled to Published", arXiv 2026

SemDN: one query, one ranked list of chunks



A chunk is a semantic unit: a text passage, a table block, an image region, a page screenshot. We focus on passages and tables.

Four mismatches between the workload and the stack

O1 · Objective

Similarity is not utility

Retrievers rank by similarity; agents need passages that help produce a correct answer.

O2 · Granularity

Pages versus chunks

The web delivers pages; a model reads a few kilobytes of passages.

O3 · Locality

URLs versus semantics

Caches key on URLs; agent queries reuse chunks that a URL cache cannot see.

O4 · Preprocessing

Redundant and failure-prone

Every operator fetches, cleans, chunks, and embeds the same pages again.

A similar passage is not always a useful one

"Tallest skyscraper in the UK?" (CRAG)

A · top by embedding similarity
"The UK had not been noted historically for its abundance of skyscrapers…"

B · top by LLM reranker
"The Shard, with a height of 309.6 meters, is the tallest skyscraper in the UK."

With A the reader answers "Landmark Pinnacle".
With B it answers "The Shard".

"Can I cancel Halden mid-contract?"

0.88 · To cancel, open Billing and choose End plan.

0.86 · §9.2: either party may terminate for material breach.

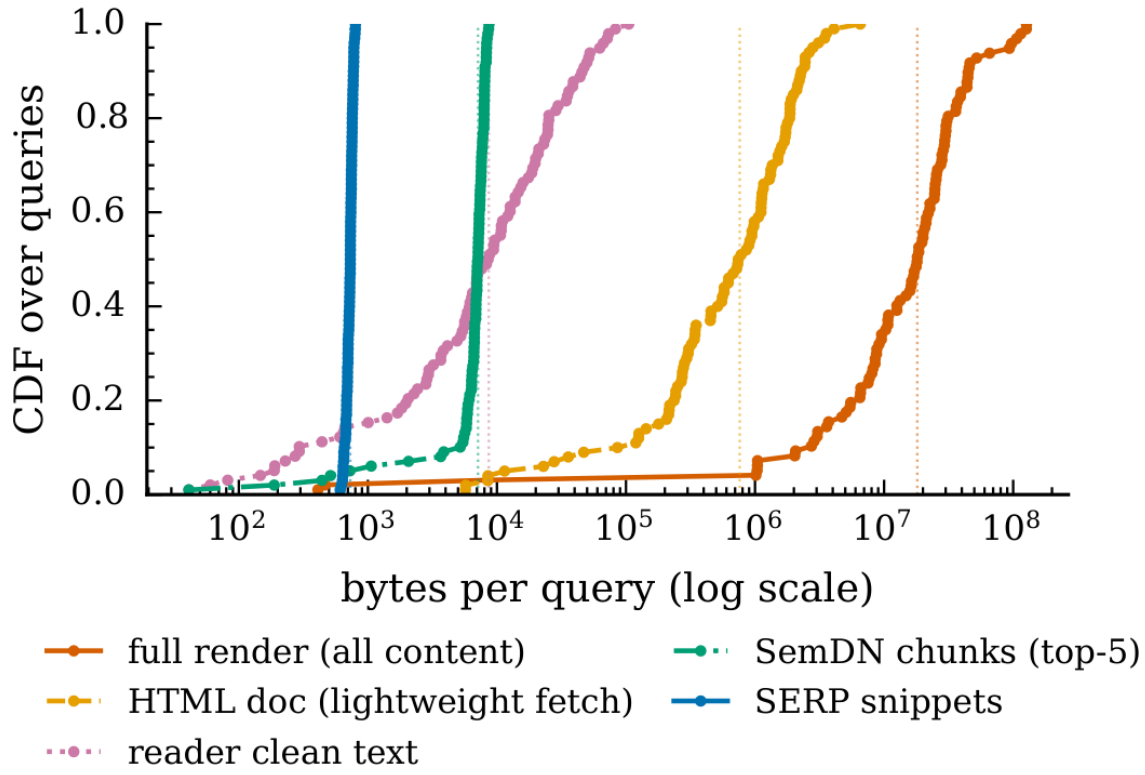
0.60 · Many customers pause rather than cancel.

A support bot wants the procedure.
A compliance agent wants §9.2.

Content and candidates can be shared. The utility ranking stays with the tenant.

Yang et al., "CRAG – Comprehensive RAG Benchmark", NeurIPS 2024 · Cuconasu et al., "The Power of Noise", SIGIR 2024 · Jin et al., "Search-R1", arXiv 2025

Pages move far more bytes than the model reads

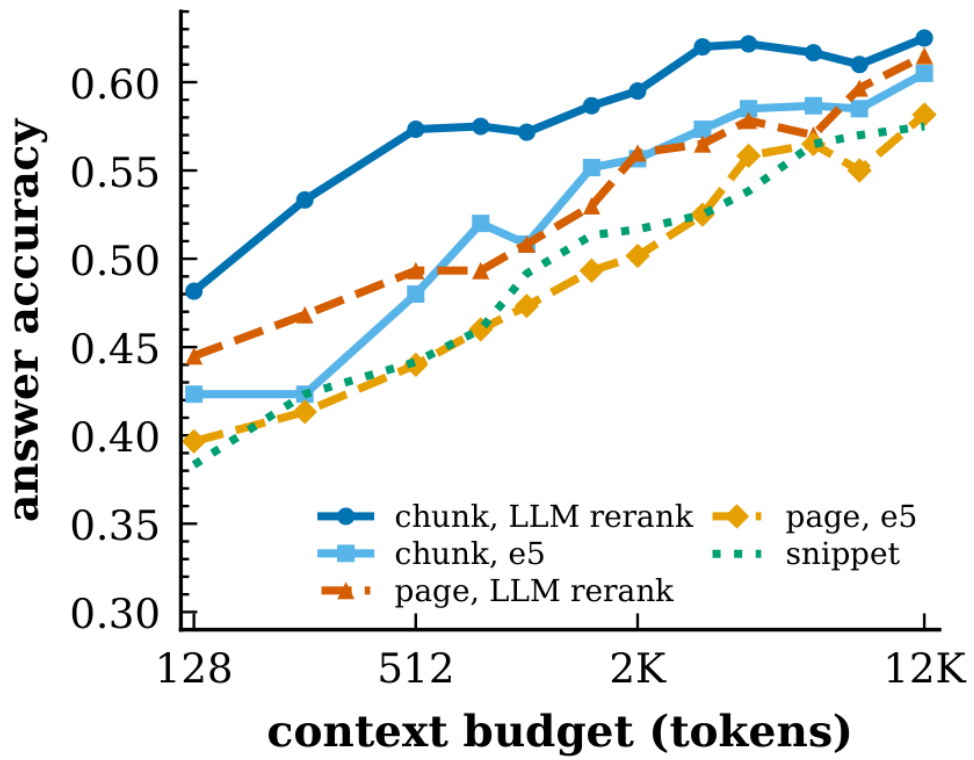


Bytes per query over the top-5 URLs, 100 live queries (Fig. 2)

Median per query	**Bytes**	**Ratio**
Chunks the model reads	7.1 KB	1×
HTML fetch, no JS	761 KB	107×
Full browser render	18.1 MB	2,532×

- ≥ 39% of 500 sampled URLs lose over half their text without JavaScript
- ≈ 19% are bot-blocked

Under a fixed context budget, chunks beat pages



CRAG Task 3, n = 600, Qwen3-32B reader (Fig. 3c)

- Chunks beat pages by 3.5–8.2 accuracy points from 256 to 6,144 tokens
- The gap narrows above 8K tokens, once context already holds the evidence

"At the Oscars, who won best actor in 2000?"

Page
starts with the 1929 winner; budget runs out before 2000

Chunk
the 2000 ceremony passage names Kevin Spacey

Yang et al., "CRAG – Comprehensive RAG Benchmark", NeurIPS 2024 · Liu et al., "Lost in the Middle", TACL 2024

Pages move more bytes and still miss evidence

Fixed corpus (wiki-18), five QA sets, Qwen2.5-7B reader; the page client downloads ten pages

Ranking	Chunks	Top pages
e5 retriever	0.456	0.427
e5 + Qwen3-Reranker-4B	0.500	0.493
Bytes delivered	3.2 KB	3.8 MB (≈1,200×)

Accuracy judged with CRAG's metric. A Qwen3-32B reader shows the same order.

Reach
16–29% of the global top-5 chunks lie on pages a whole-page ranker does not select, even with 20 pages downloaded.

Chunks match or beat pages while moving three orders of magnitude fewer bytes.

Yang et al., "HotpotQA", EMNLP 2018 · Kwiatkowski et al., "Natural Questions", TACL 2019 · Yang et al., "CRAG – Comprehensive RAG Benchmark", NeurIPS 2024

URL caches miss chunk reuse; operators repeat the work

O3 · Locality

- A search agent draws 70–100% unseen URLs within a task
- Yet 32–55% of a new query's chunks were already retrieved earlier in the same task
- That reuse is invisible to a cache keyed on URLs

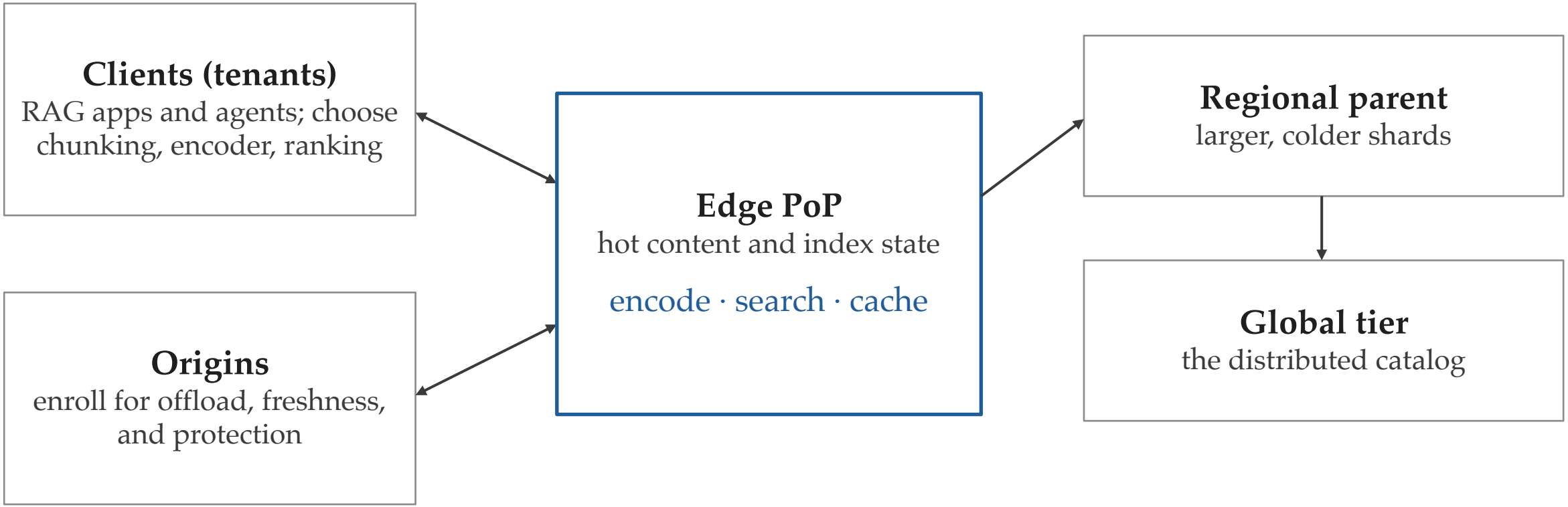
O4 · Preprocessing

- Across 1,430 live SERP URLs, a cloud renderer gets usable text from at most 75%
- Stack Overflow and Server Fault returned 403 to a plain fetch
- One YouTube page: 1.4 MB of HTML, 147 bytes of clean text

Origins absorb the resulting fetch storms, once per operator.

Zhang et al., "Rethinking Web Cache Design for the AI Era", SoCC 2025 · Cloudflare, "Browser Rendering", 2026 · Fastly, "Q2 2025 Threat Insights Report", 2025

Two kinds of customers, an edge hierarchy, three functions



Encode queries and content · **Search** canonical or registered indexes · **Cache** blocks, chunks, and index state

Aly et al., "Taily: Shard Selection Using the Tail of Score Distributions", SIGIR 2013

Keying delivery on queries changes three things

Misses

CDN: A URL lookup misses on an exact name.

SemDN: Top-k search always answers, so missing or stale evidence must be estimated.

Tenancy

CDN: Origins still enroll for offload and protection.

SemDN: Each client chooses the encoder and index it retrieves with, a choice origins need not see.

State

CDN: A cached page serves every client.

SemDN: Index and cache state serve only clients sharing a representation; acquisition and normalization are shared by all.

SemDN could be built on an existing CDN or as a separate network. Either way, it is more than finer-grained caching.

One design decision per mismatch

O1 **Share content and candidates; rank per tenant**

Tenant utility functions and labels stay private

O2 **Serve and cache at chunk granularity**

Return passages, not pages; cache at the granularity the workload reuses

---

O3 **Place and evict in the query stream**

The edge sees semantic reuse, not only URL popularity

O4 **Shared work once; specialized work once per representation**

Acquisition and normalization run once per content version

Top-k search has no miss bit

Three actions: serve the chunks on hand (cost 0) · ask the regional parent (1) · search externally (5)

Query	What the edge holds	Right action
"Current free-tier rate limit for the Halden API?"	Three chunks agree: 60 requests per minute; 4–7 months old	**Search: the limit dropped to 30 nine days ago**
"What does HTTP status 429 mean?"	Three chunks agree: too many requests; 2–3 years old	**Serve: old, but the meaning has not changed**
"Opening stock price of Impinj today?"	Chunks are 2–3 days old	**Search: "today" makes any cached price stale**

Agreement among chunks and their age do not decide the miss. The query does.

Yang et al., "CRAG – Comprehensive RAG Benchmark", NeurIPS 2024

A coverage controller estimates the risk of a semantic miss

Semantic coverage miss: the estimated risk that external discovery would add fresh evidence and change the returned top-k exceeds a policy threshold.

Imperfect signals of coverage risk

- Score distribution and result coherence
- Sufficiency: does the evidence hold enough to answer?
- Agreement with the regional parent and global tier
- Timeliness metadata on each block

Open problems

- Edge miss versus system-wide miss
- False confidence hides fresh evidence; caution restores redundant search
- Calibration under drift, such as a breaking-news week

Suting Chen*, Peichun Hua*, and Yunming Xiao. "Relevance Is Not Sufficient Evidence", arXiv 2026
Yan et al., "Corrective RAG", arXiv 2024 · Mansoor et al., "Freshness-Aware Semantic Caching", arXiv 2026

Three kinds of chunk reuse within a task

Later turns of a task, top-3 chunks shown, byte-identical repeat queries removed

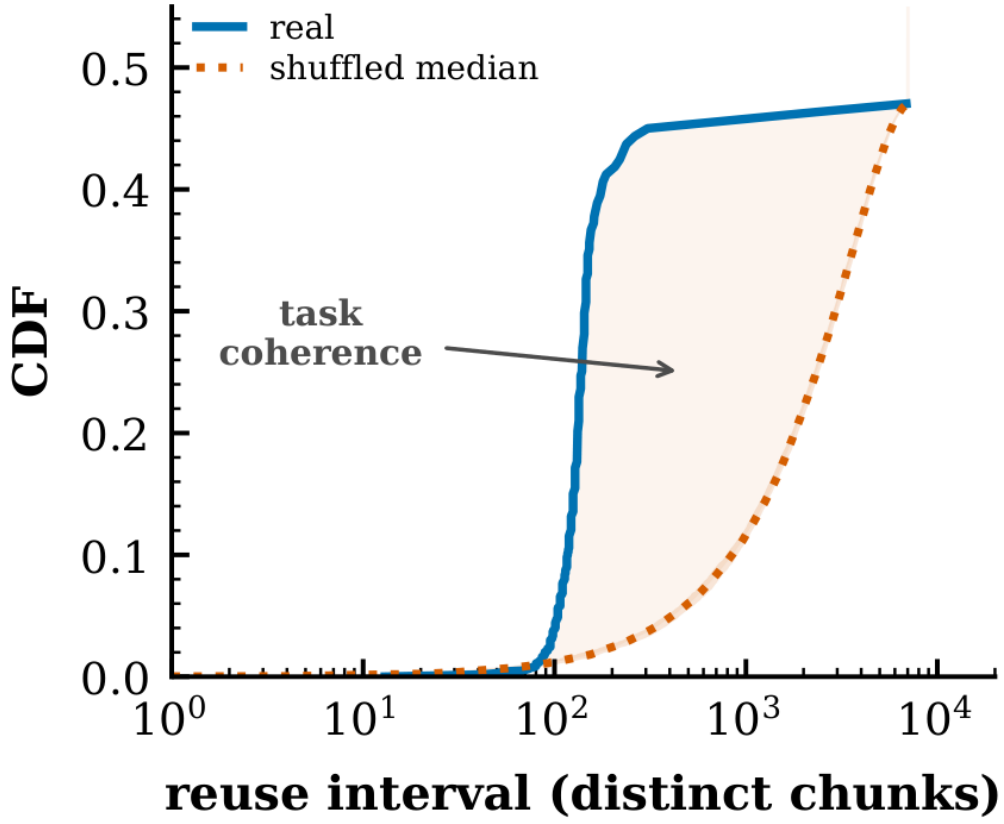
	Search-R1 / wiki-18	**CRAG**
Exact repeat: a chunk the agent already saw	41% (32% → 55% by turn)	70%
Sibling: unseen chunk on a page already fetched	11%	10%
New page	48%	20%

Cut-off: an earlier search returned the chunk but did not show it. Of chunks first appearing in the next turn's top 3, 25% were in ranks 4–13 of the search just served, and 38.5% in ranks 4–50.

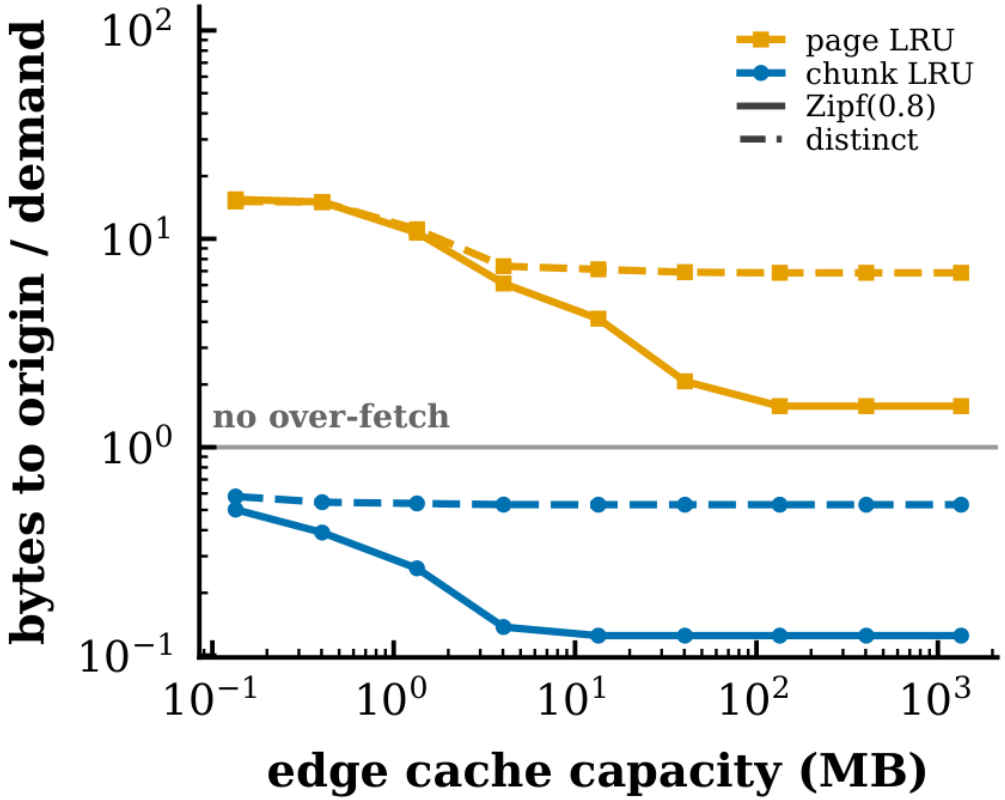
Traces: Search-R1 with Qwen2.5-7B, an older agent. We expect stronger effects with a better agent and retriever.

Jin et al., "Search-R1", arXiv 2025 · Yang et al., "CRAG – Comprehensive RAG Benchmark", NeurIPS 2024

Reuse is task-driven, and a small chunk cache captures it



Reuse comes 10–100× sooner than in task-shuffled controls (Fig. 3a)



Chunk cache: 13–41× fewer byte traffic to origin than a URL cache (Fig. 3b)

Keeping chunks fresh, and how the query stream can mislead

Three ways to refresh a block

- **Per-block TTL:** stale windows between refreshes; wasted fetches on stable pages
- **Query piggyback:** one re-fetch refreshes its siblings; cold content never gets warmed
- **Origin purge feed:** exact, but only for enrolled origins

Three ways the query stream misleads

- **Coverage skew:** a rare-disease reference queried once a month falls out of the edge
- **Cross-tenant leakage:** one tenant's hot subset shows up as timing to others
- **Feedback entrenchment:** a passage with a long record of correct answers survives a rule change

Likely answer: a hybrid. Savings must be estimated, not assumed.

Mansoor et al., "Freshness-Aware Semantic Caching", arXiv 2026 · Gu et al., "Auditing Prompt Caching in Language Model APIs", ICML 2025

What retrieval state is shareable across tenants?

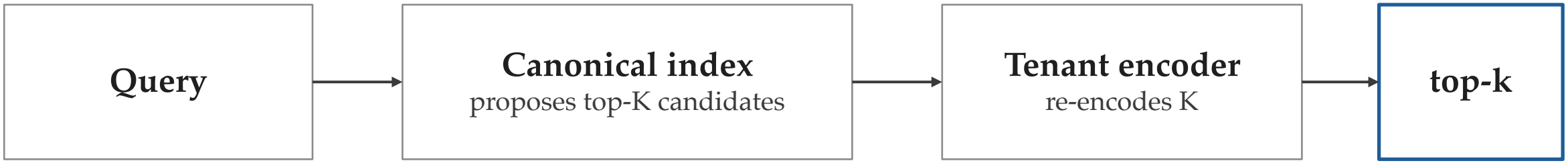
Acquisition and cleaning	shared by all tenants
Normalized, versioned blocks	shared by all tenants
Canonical embeddings and index	shared by tenants on the canonical representation
Registered model-specific index	shared among tenants registered for that model
Private encoder	one tenant; embedding work not amortized

Modeled amortization

Measured costs fed into a Zipfian demand model: at 50 tenants, about 6× less repeated fetch-clean-chunk-embed work. A private encoder stays at 1×.

Pricing can steer this: discount the shareable canonical representation.

Serving many encoders without many indexes



k ≪ K. Hot chunks' re-encoded vectors are cached and grow into a model-specific index where demand justifies it.

Open trade-offs

- How candidate recall degrades with encoder drift
- When re-encoding stops paying off

Storage versus compute

- Recompute-based indexes store few embeddings and pay GPU time
- Whether real PoPs carry that compute is open

Wang et al., "LEANN: A Low-Storage Vector Index", arXiv 2025 · Seemakhupt et al., "EdgeRAG", arXiv 2024 · Hua and Xiao, "Matryoshka Hash Representations", arXiv 2026

A two-sided service: who pays for what



- Buys offload for machine traffic
- Sets transformations, freshness, purge, provenance

- Reverse proxy for origins, forward proxy for clients
- Neither side pays just because the other uses it

- Buys retrieval quality and latency
- Billed per chunk consumed
- Chooses encoder and chunking

Bootstrapping: SemDN acquires public pages itself; client traffic shows which origins to enroll.

Cloudflare, "An AI Index for All Our Customers", 2025 · Cloudflare, "AI Search", 2026

Why should anyone trust the edge?

A CDN is a courier: it delivers the site's page sealed, and HTTPS shows the seal is intact. SemDN is a librarian: it cuts pages into passages and picks which ones you read.

Is this what the site said?

Origins sign blocks; each chunk traces to the site's text and version

HTTP message signatures, Merkle proofs

Did I get the best passages?

The edge commits to its index and proves the top-k is honest

verifiable vector search; pay-to-rank bias

Was anyone left out?

A public list of enrolled origins

like Certificate Transparency logs

The catch is cost. Verifiable search takes 0.8–2 s per query. Spot-checking a sample catches a cheating edge, but still adds hundreds of milliseconds per answer.

Zou et al., "PoisonedRAG", USENIX Sec. 2025 · Backman et al., "HTTP Message Signatures", RFC 9421 · Qiu et al., "V3DB", arXiv 2026 · Avramov et al., "Atlas", arXiv 2026

Make semantic selection a network-delivery service

- Agents consume passages; today's stack discovers and delivers URL-addressed pages
- SemDN joins a search engine's query interface with a CDN's origin-authorized edge delivery
- Early evidence: pages move orders of magnitude more bytes; chunk reuse is invisible to URL caches; chunks beat pages per token

Open questions

1. When is cached evidence good enough to serve, given timeliness and coverage?
2. What belongs in a client cache, and what must be shared to justify a network?
3. Where should the encoder boundary sit, and when does re-encoding stop paying off?
4. How much query state can tenants share without exposing their tasks?
5. How do origins and clients enroll first, and how does the edge serve origins that refuse?
6. Can clients verify the best chunks were served, without seeing the index?
7. How does one query find its sources across millions of sites?

References

[1] Peichun Hua and Yunming Xiao. Semantics Delivery Network: Rethinking Web Retrieval Infrastructure for LLM Agents. HotNets 2026. arXiv:2609.22486.

[2] Patrick Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.

[3] Bowen Jin et al. Search-R1: Training LLMs to Reason and Leverage Search Engines with Reinforcement Learning. arXiv:2503.09516, 2025.

[4] Fastly. Q2 2025 Threat Insights Report: The Rise of AI Crawlers and Fetchers. 2025.

[5] Cloudflare. An AI Index for All Our Customers. Blog, 2025.

[6] Cloudflare. Cloudflare AI Search. Developer documentation, 2026.

[7] Yixi Chen et al. From Crawled to Published: A Publisher-Controlled Web Architecture for an AI-Native Internet. arXiv:2511.18354v2, 2026.

[8] Xiao Yang et al. CRAG – Comprehensive RAG Benchmark. NeurIPS 2024.

[9] Florin Cuconasu et al. The Power of Noise: Redefining Retrieval for RAG Systems. SIGIR 2024.

[10] Nelson F. Liu et al. Lost in the Middle: How Language Models Use Long Contexts. TACL 2024.

[11] Zhilin Yang et al. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. EMNLP 2018.

[12] Tom Kwiatkowski et al. Natural Questions: A Benchmark for Question Answering Research. TACL 2019.

[13] Yazhuo Zhang et al. Rethinking Web Cache Design for the AI Era. ACM SoCC 2025.

[14] Cloudflare. Browser Rendering: Markdown endpoint. Developer documentation, 2026.

References (continued)

[15] Robin Aly et al. Taily: Shard Selection Using the Tail of Score Distributions. SIGIR 2013.

[16] Suting Chen, Peichun Hua, and Yunming Xiao. Relevance Is Not Sufficient Evidence: Detecting Evidence Gaps Before Generation in RAG. arXiv:2609.37469, 2026.

[17] Shi-Qi Yan et al. Corrective Retrieval Augmented Generation. arXiv:2401.15884, 2024.

[18] Muhammad Mansoor et al. Risk-Constrained Freshness-Aware Semantic Caching for Open-Web Retrieval-Augmented LLMs. arXiv:2607.04281, 2026.

[19] Seungjin Lee et al. Characterizing Locality in Large-Scale Vector Search Workloads of Agentic AI Systems. VLDB 2026 Workshop on Vector Databases.

[20] Chenchen Gu et al. Auditing Prompt Caching in Language Model APIs. ICML 2025.

[21] Yichuan Wang et al. LEANN: A Low-Storage Vector Index. arXiv:2506.08276, 2025.

[22] Korakit Seemakhupt et al. EdgeRAG: Online-Indexed RAG for Edge Devices. arXiv:2412.21023, 2024.

[23] Peichun Hua and Yunming Xiao. Matryoshka Hash Representations for Model-Aware Compact Semantic Retrieval. arXiv:2609.07276, 2026.

[24] Wei Zou et al. PoisonedRAG: Knowledge Corruption Attacks to Retrieval-Augmented Generation of Large Language Models. USENIX Security 2025.

[25] Annabelle Backman, Justin Richer, and Manu Sporny. HTTP Message Signatures. RFC 9421, 2024.

[26] Zipeng Qiu et al. V3DB: Audit-on-Demand Zero-Knowledge Proofs for Verifiable Vector Search over Committed Snapshots. arXiv:2603.03065, 2026.

[27] Nikolay Avramov et al. Atlas: Efficient Verifiable Semantic Search. arXiv:2609.11841, 2026.

[28] Ben Laurie, Eran Messeri, and Rob Stradling. Certificate Transparency Version 2.0. RFC 9162, 2021.