Reliable Context

RELIABLE CONTEXT / RESEARCH & PRACTICAL GUIDANCE

Cached answer validity test: is a cached answer still valid?

> Reliable Context is a kernel of truth: a system of record for organizations that fuse private and public information and must defend every value they act on. One canonical record per real thing, a cited source and an observation date behind every value, unknowns declared rather than left blank, and any past state of the record reproducible.

Published by Reliable Context · 2026-10-20 · cached answer validity test · semantic cache correctness · cache invalidation for ai answers · stale data risk · data provenance and lineage · reuse decision ledger · false reuse rate · semantic caching · answer reuse validity

Original article: https://reliablecontext.com/articles/cached-answer-reuse-validity-test



In this article

1. Executive Summary
2. Introduction and Background
3. Definitions and the Two Proof Obligations
4. Key Changes From Similarity-Based Reuse to Validity-Based Reuse
5. Implementation Considerations and Process Changes
6. Data Analysis and Evidence
7. Implications and Future Directions
8. Frequently Asked Questions (FAQs)
9. Conclusion

Executive Summary

A cached answer is valid for a new request only when two separate things are shown: the new request asks the same proposition the answer was built for, and the evidence behind the answer is still eligible for this request, this reader and this moment. High query similarity establishes neither. Redis documents the trade openly: semantic caching "gives up the guarantee that a cache hit is always correct" (redis.io), and its own documentation frames the threshold as a dial between wrong answers and hit rate (redis.io). A peer-reviewed study published on **11 February 2026** describes caches that depend only on semantic similarity as vulnerable to adversarial exploitation (www.nature.com).

This report proposes an acceptance contract rather than an implementation recipe: a **paired reuse matrix** that crosses request meaning (literal repeat, faithful paraphrase, changed entity, changed date, reversed negation, different unit or comparison class, changed requested action) with evidence state (unchanged, factual correction, retraction, unavailable source, new exception, changed reader eligibility). The correct outcome for every cell is written down before any cache or verifier is run. A paraphrase over unchanged support may pass. A nearby question with a changed date, entity or exception must fail. An unchanged old capture can still defend the historical answer, but it does not by itself justify serving that answer today.

The metrics are deliberately separated. **False-reuse rate** is accepted invalid candidate answers divided by all accepted candidates, and it is undefined, not perfect, when nothing was accepted. **Invalid-candidate acceptance** and **valid-reuse recall** are reported next to it, with raw counts and intervals that respect request families ((cameron.econ.ucdavis.edu)). Failures from question mismatch, support invalidation and reader scope are tallied separately and never summed as unique failures when reasons overlap.

Published thresholds are starting points, not guarantees. Redis lists a default similarity threshold of **0.85** with a suggested 0.8 to 0.9 starting range (redis.io), a Microsoft policy page warns that a score threshold above **0.2** on its scale may cause cache mismatch (learn.microsoft.com), and the Scientific Reports authors state that their best threshold is specific to one embedding model (www.nature.com). As of **October 2026**, no public benchmark was found that crosses request meaning with evidence state. The evaluation data of the cited paper are available only on reasonable request (www.nature.com), so independent replication is not established.

Introduction and Background

When a new request arrives, must the full evidence evaluation run again, or can an answer generated earlier be served? The decision looks like a cache problem, but the hard part is not storage or speed. It is information equivalence (does the new request ask what the old one asked) and evidential validity (does the support for the old answer still hold, for this reader, now).

Web caching standardized the evidence half of the problem long ago. The HTTP caching specification forbids reusing a stored response unless a list of conditions holds (www.rfc-editor.org), requires revalidation unless the headers named by Vary match the original request (www.rfc-editor.org), and distinguishes a fresh response from a stale one by age against a freshness lifetime (www.rfc-editor.org). Semantic caching for large language model (LLM) applications addresses the question half by matching paraphrases, and vendors that document it are candid about the cost. Microsoft's policy documentation notes that similarity-based reuse can surface responses that are "incorrect, outdated, or unsafe for the current request" (learn.microsoft.com).

What is missing from public material is a portable, reviewable test that joins the two halves. A similarity threshold decides whether two request strings are close. It does not decide whether the cached answer's evidence remains eligible. This report defines the test, specifies the measurements, and states what is and is not established. It is about **information equivalence and evidential validity**, not cache performance, not any sector, and not the architecture of any product. Where a reader needs a semantic cache, that software category is the right tool for the lookup step, and nothing here proposes replacing it. A system of record, which keeps one cited value with an observation date per fact, answers a different question: what the evidence was and when it was observed. The acceptance contract below sits between them.

All fixtures described are synthetic or explicitly authorized test data. Product mechanisms, legal conclusions and certifications are out of scope, and this is not legal advice.

Definitions and the Two Proof Obligations

- **Answer reuse:** serving a stored, previously generated final answer for a new request without repeating the evidence evaluation that produced it.
- **Request meaning:** the proposition, scope and requested action a request expresses, independent of its wording. Two requests are equivalent only if the answer to one is a correct answer to the other.
- **Evidence eligibility:** whether a specific source version may support an answer to this request for this reader at this time, taking into account correction, retraction, availability and access policy.
- **Current support:** evidence that is eligible now. **Historical support:** evidence that was eligible when the answer was produced and remains a faithful record of that moment.
- **Observation time (observed_at):** when the system looked at the source. **Valid time:** the period during which a statement holds in the modeled world.

Response caching is not prompt-prefix or KV caching

Three different things are called caching in LLM stacks, and only one of them stores answers. Anthropic's prompt caching reuses a prompt prefix, requires identical prompt segments for a hit (platform.claude.com), and states that it has "no effect on output token generation" (platform.claude.com). OpenAI's guide likewise

requires the entire rendered prefix to match and says the model still processes new input to generate a new response (developers.openai.com). At the serving layer, vLLM's automatic prefix caching stores the key-value (KV) cache of earlier queries for reuse when a new query shares a prefix (docs.vllm.ai), and it speeds up only prompt processing, not generation (docs.vllm.ai). None of these return a stored final answer, so none raise the validity question studied here. Only response caching, exact or semantic, does.

Two proof obligations

A cached answer carries two independent obligations:

1. **Question obligation:** the new request means what the original request meant, within the same entity, time, qualifier, unit and requested action.
2. **Evidence obligation:** the support recorded for the answer is still eligible for this request, reader and time.

The obligations fail independently. A literal repeat satisfies the first and may fail the second once the source is corrected. A request about a neighboring entity fails the first even when the source never changed. The evidence obligation splits further by time. An unchanged old capture is excellent historical support: the Memento specification defines a memento as a resource that encapsulates the state the original resource had at time T (www.rfc-editor.org) and treats the datetime header as a promise that the archived state will not change (www.rfc-editor.org). That promise concerns the archived state, not whether the live source still says the same thing. Historical support defends what was concluded then; only current support defends serving it today.

Time has two axes

Temporal database research separates when something is true from when it was recorded. Kulkarni and Michels define valid time as the period during which a row is regarded as correctly reflecting reality ((sigmodrecord.org)) and transaction time as the period during which a row is committed to the database ((sigmodrecord.org)). Fowler describes the same split as actual history and record history (martinfowler.com). A cached answer sits on this seam: its evidence has a valid time (the date the proposition concerns) and an observation time (when the system looked). A changed date moves along the first axis; a source correction moves along the second.

Key Changes From Similarity-Based Reuse to Validity-Based Reuse

Change 1: from a similarity score to proposition equivalence

Similarity measures closeness in an embedding space, not whether an edit preserves meaning. The vCache authors report that similarity values of correct and incorrect reuse exhibit highly overlapping distributions, and that static thresholds do not give formal correctness guarantees (arxiv.org). The category-aware caching paper argues that a fixed threshold produces false positives in dense embedding spaces and misses valid paraphrases in sparse ones (arxiv.org).

The **NevIR** benchmark asks retrieval models to rank two documents that differ only by negation (arxiv.org) and reports that models generally do no better than a random ranking (arxiv.org). A diagnostic study finds that most unsupervised sentence encoders are insensitive to negation (arxiv.org), and shows that two encoders with similar scores on standard similarity benchmarks can behave very differently on minimal pairs (arxiv.org). A 2025 study finds that 13 widely used text embedding models generally struggle with numerical details (arxiv.org). Taken together, negation, numbers and entity swaps are where a paraphrase detector based on a global score is least reliable, which is the reason the matrix includes them. These are studies of embeddings in general, not of any specific product.

The Scientific Reports paper labels its semantic perturbations as entity substitution, numeric perturbation, polarity inversion and category substitution (www.nature.com). Those labels map closely onto the meaning-changing request classes below. The reader should note what the paper's headline number is. It reports adversarial attack success falling from **52.77% to 14.27%** for the authors' proposed design relative to a single-query similarity baseline (www.nature.com), but defines success at cluster level, where a single adversarial infiltration counts as a success (www.nature.com). The runtime figure is separate: **8.0%** at the default centroid threshold (www.nature.com). Neither number is the share of live requests served a wrong cached answer, and the authors caution that their results should not be interpreted as formal guarantees (www.nature.com).

Change 2: from an expiry time to evidence eligibility

Time-to-live (TTL) is the standard freshness control, and it is useful. Azure's store policy requires a duration in seconds (learn.microsoft.com), and AWS describes invalidation as updating or removing cached entries when the underlying data changes (aws.amazon.com). But a TTL is a statement about elapsed time since storage. It is not a statement about the source. The Redis service settings documentation says that with no TTL configured, items remain until manually removed (redis.io). Expiration alone does not prove that a source did not change during the entry's lifetime, and these documented controls are not guarantees about source state. The additional obligations proposed here are ours.

HTTP caching shows the missing piece. It separates freshness (age against lifetime) from validation, where a conditional request lets the origin confirm that a stored response is still valid (www.rfc-editor.org). Validators carry strength: a strong validator must change whenever the representation changes in an observable way (www.rfc-editor.org), while a weak validator may stay the same across some changes (www.rfc-editor.org), and a Last-Modified time is treated as weak unless specific rules show otherwise (www.rfc-editor.org). A cached answer needs the equivalent of a strong validator over its evidence: a source version identifier, not merely a timestamp.

Standards also record deliberate exceptions: the stale-if-error extension allows a stale response to be served when an error occurs ((www.rfc-editor.org)). For answer reuse the analogue is a declared policy, never a silent default, and the ledger must record that it was applied.

Change 3: from one accuracy number to separated failure modes

A single hit-correctness figure conceals why a reuse was wrong. The proposal separates three reasons and allows them to overlap on a single request:

- **Question mismatch:** the cached answer addresses a different proposition.

- **Support invalidation:** the question matches but the evidence is corrected, retracted, unavailable or carries a new exception.
- **Reader-scope failure:** question and evidence are fine, but this reader was not eligible for the evidence when the answer was produced or is not eligible now.

Redis's documentation points toward hard boundaries for the third category. It states that effective semantic caching combines soft similarity matching with hard metadata boundaries (redis.io), and LangCache search returns only responses whose attributes match those specified (redis.io). Azure's lookup policy offers a vary-by expression to control cross-user access (learn.microsoft.com). These scoping mechanisms can be exercised by a test, but an access filter is not evidence that no information leaks behaviorally through shared responses.

The metrics map onto standard classification terms. Precision is the ability of a classifier not to label as positive a sample that is negative (scikit-learn.org), which corresponds to the false-reuse rate's complement, and recall is the ability to find all the positive samples (scikit-learn.org), which corresponds to valid-reuse recall. The two trade off against each other (nlp.stanford.edu), so both must be reported.

Change 4: from candidate selection to candidate selection plus validity checking

If only accepted hits are inspected, unsafe candidates that were rejected by luck or by a later rule are invisible, and so is the quality of the threshold. The evaluation must therefore retain both stages. Candidate selection (which stored answers are proposed for this request) is measured on every tested pair. Validity checking (whether a proposed candidate may be served) is measured separately. Systems that add a verification stage exist in the literature: **Krites**, for example, adds an asynchronous LLM-judge check outside the critical path (arxiv.org), and **vCache** learns per-prompt thresholds to meet a user-set maximum error rate (arxiv.org). The vCache authors state that their guarantee relies on independent, identically distributed data and a sigmoid function family (arxiv.org), which is why a fixture with adversarially chosen near-miss requests remains informative.

Change 5: from a served hit to a disposition on a ledger

An answer that was served once has no authority from that fact. Redis describes its semantic cache as non-authoritative and rebuildable at any time (redis.io). The consequence is that an entry with unknown support must not be promoted to an authoritative record merely because it was served earlier. Provenance research supplies the vocabulary for the record that should exist instead. **Buneman, Khanna and Tan** define data provenance as where a piece of data came from and the process by which it arrived (link.springer.com), and tie it to the accuracy and currency of data (link.springer.com). The W3C PROV model defines derivation as a transformation or update of an entity yielding a new entity (www.w3.org) and invalidation as the start of the expiry of an existing entity (www.w3.org), with properties for the time an entity was generated (www.w3.org) and invalidated (www.w3.org). Applied here, the cached answer is an entity derived from evidence entities, and a correction invalidates the evidence at an observable time.

Implementation Considerations and Process Changes

The protocol below is an observable acceptance contract. It specifies what to record and measure, not how a cache or verifier is built.

Step 1: versioned synthetic corpus and seed answers

Build a small synthetic corpus with version identifiers, so every evidence state can be produced on demand. For each seed answer, store a **cached-answer evidence record**:

- **Answer ID** and the exact claims made, one proposition per claim.
- **Original source and verbatim span** supporting each claim.
- **Source version** and **observed_at** for each span.
- **Valid-time scope:** the date or period the proposition concerns.
- **Qualifier and entity scope:** which entity, unit, exception and comparison class make the answer valid.
- **Eligibility snapshot:** which reader scopes could see the evidence when the answer was produced.
- **Permitted reuse conditions:** what changes require re-evaluation.

Provenance tooling already models some of these fields. The OpenLineage specification models a run as one occurrence in time of a job (openlineage.io) and supports a dataset version facet when the store defines versioning (openlineage.io).

Step 2: paired requests

For each seed answer, author paired requests in two groups. Meaning-preserving requests are a literal repeat and a faithful paraphrase. Meaning-changing requests are a changed entity, a changed temporal question, a reversed negation, a different unit or comparison class, and a changed requested action. Several are close in wording but demand a different answer. Every pair is labeled by a reviewer, with a rationale, before any system sees it. *Table 1* shows the **request-equivalence table** with synthetic content.

Original request	New request	Preserved proposition	Altered scope	Verdict	Evidence required	Reviewer rationale
What is the upload limit of Plan A as of 2026-09-01?	What is the upload limit of Plan A as of 2026-09-01?	Same	None	Equivalent	Current support for the same span and version	Literal repeat, only the evidence obligation remains
What is the upload limit of Plan A as of 2026-09-01?	As of 1 September 2026, how large may an upload be on Plan A?	Same	None	Equivalent	Current support, same span	Faithful paraphrase, same entity, date, unit
What is the upload limit of Plan A as of 2026-09-01?	What is the upload limit of Plan B as of 2026-09-01?	Different	Entity	Not equivalent	Evidence for Plan B	Neighboring entity, wording nearly identical
What is the upload limit of Plan A as of 2026-09-01?	What is the upload limit of Plan A as of 2026-12-01?	Different	Valid time	Not equivalent	Evidence whose valid time covers 2026-12-01	Date changed, the old span may not cover it
Which plans include audit export?	Which plans do not include audit export?	Complement	Polarity	Not equivalent	Evidence for the full plan list	Reversed negation, the answer is the complement

Original request	New request	Preserved proposition	Altered scope	Verdict	Evidence required	Reviewer rationale
What is the upload limit of Plan A in megabytes per file?	What is the upload limit of Plan A in megabytes per account?	Different	Unit and comparison class	Not equivalent	Evidence stating the per-account limit	Same number cannot be assumed across units
What is the upload limit of Plan A?	Approve this 900 MB upload on Plan A.	Different	Requested action	Not equivalent	Limit plus exception rules for this reader	A decision, not a fact lookup

Table 1 is a template with invented content (Hypothetical Example). The verdict column is set by a human reviewer and frozen before evaluation. The equivalence label is an input to the test, never an output of the system under test.

Step 3: cross requests with evidence states

Cross each request class with evidence states: unchanged evidence, factual correction, retraction, unavailable source, a new exception, and changed evidence eligibility for the reader. The expected outcome of each cell is specified before the cache or verifier is run. A fresh capture and an old capture have distinct support roles: the fresh capture can license current reuse, the old capture can only license the historical claim. *Table 2* gives the grid, using five outcome codes: **S** (serve the cached answer), **E** (re-evaluate evidence before serving), **H** (hold: do not serve, record that the source is blocked or not found, with a reason), **M** (miss: this is a different proposition, evaluate it on its own), and **X** (do not serve to this reader).

Request class	Unchanged evidence	Factual correction	Retraction	Source unavailable	New exception	Reader eligibility changed
Literal repeat	S	E	E	H	E	X
Faithful paraphrase	S	E	E	H	E	X
Changed entity	M	M	M	M	M	M
Changed temporal question	M (old span may be reused as input if its valid time covers the new date)	M	M	M	M	M
Reversed negation	M	M	M	M	M	M
Different unit or comparison class	M	M	M	M	M	M
Changed requested action	M	M	M	M	M	M

Table 2 reads as follows. Only two rows can ever produce S, and only in one column. Every other cell is a refusal to reuse, but for different reasons: E and H record an evidence-state reason, M records a question-mismatch reason, and X records a reader-scope reason. A cached answer for a meaning-changing request is never served, even with unchanged evidence, because the first proof obligation fails. Reusing evidence as input to a new evaluation is a different act from reusing the answer. The H column insists on a reason: unknown source state, blocked access and missing evidence each need their own explanation, and a cache miss is not a factual negative.

On permissions, the X column prevents new reuse for a reader whose eligibility changed. Whether the historical audit can still reconstruct what was served to whom at the earlier time is a separate policy question for the organization. The ledger should keep the eligibility snapshot so reconstruction is possible without making the old answer servable.

Step 4: freeze components and evaluate four arms

Evaluate the same workload under four arms: cache bypassed, exact-response reuse, candidate semantic reuse, and the declared validation policy applied on top of candidate selection. Freeze generator settings and evidence snapshots. Vary one component at a time, since changing the embedding model, threshold and verifier together makes the result unattributable. The thresholds are starting suggestions, not acceptance guarantees: Redis recommends starting conservative and tighter (redis.io) and notes that a support FAQ bot can tolerate a looser threshold than an account-specific financial bot (redis.io). Redis documents an exact search strategy that returns a result only when the stored prompt matches exactly (redis.io), a natural exact-reuse arm.

Step 5: check which context affects candidate eligibility

A single question string cannot establish all contexts. For the system under test, verify with controlled pairs whether tenant, locale, model version, permissions and prior conversation turns influence candidate eligibility. Azure's lookup policy exposes a message-count setting after which caching is skipped ((learn.microsoft.com)), and **MeanCache** treats context as a separate source of wrong hits by encoding context chains for each cached query ((arxiv.org)). Whether a product includes conversation history in its lookup key must be read from its current documentation and then tested. The LangCache concepts and service-creation pages reviewed for this report did not state whether prior turns are part of the key.

Step 6: the lookup-to-serving boundary test

Source and policy can change between lookup and serving. Treat this as a separately specified boundary test: in an instrumented fixture, change a source version or policy at a declared time while serving continues, and record every serve with a timestamp. The measurement is the **change exposure time**, the last observed unsafe serve time minus the declared change time. Report the sampling resolution and the censored cases (no unsafe serve observed before the run ended). Censoring has a standard treatment in reliability statistics: NIST defines censored data as the case where not all units on test fail (www.itl.nist.gov), and units that survive the whole fixed-length test are right-censored observations (www.itl.nist.gov). A finite polling interval can bound exposure but cannot supply instantaneous truth, so the claim to publish is an exposure window measured from

declared observation points. Bulk invalidation shrinks the window but is not itself a validity check: LangCache can delete entries that contain all given attributes (redis.io), which presupposes that entries were tagged at write time with the dimension that later changes.

Step 7: publish a reuse decision ledger and re-evaluation triggers

Every accepted reuse writes a ledger row: request, candidate answer ID, equivalence verdict and rationale, source version and observed_at checked, eligibility snapshot, disposition (served, re-evaluated, held with reason, missed, not served to reader), and the policy in force. NIST's generative AI profile suggests documenting each instance where content is generated, modified or shared to enable traceability (nvlpubs.nist.gov), and tracking dataset modifications for provenance (nvlpubs.nist.gov). The NIST AI Risk Management Framework adds that documentation can improve human review processes (nvlpubs.nist.gov). These are voluntary guidance documents, not a compliance claim.

Re-evaluation triggers should be explicit: a source version change, a correction or retraction, an unavailable source, a new exception, a change in reader eligibility, and a change in generator configuration.

Data Analysis and Evidence

What published figures can and cannot support

Table 3 lists published figures relevant to the decision, with the condition under which each was reported. The purpose is to show what kinds of numbers exist, not to rank systems; provider documentation figures describe settings, not independent performance results.

Source	What was reported	Condition and limit
Redis LangCache documentation, as of October 2026	Default similarity threshold 0.85, suggested starting range 0.8 to 0.9 ((redis.io))	Provider setting. Documentation states the right value depends on the cost of a wrong answer
Microsoft Azure API Management policy documentation	Score threshold above 0.2 may lead to cache mismatch ((learn.microsoft.com))	Different scale (lower means stricter), so values are not comparable with Redis
Afiffy, Fakhr and Maghraby, Scientific Reports, 11 February 2026	Attack success 52.77% to 14.27% for the proposed design versus a similarity-only baseline (www.nature.com)	Cluster-level contamination in a simulated attack, two sets of 300 intents each (www.nature.com); data on reasonable request (www.nature.com)
Same paper	8.0% runtime centroid-matching success ((www.nature.com))	Specific to the embedding model used ((www.nature.com))
vCache, Schroeder et al., ICLR 2026	Per-prompt learned thresholds meet specified error bounds while outperforming static thresholds ((arxiv.org))	Guarantee relies on i.i.d. data and a sigmoid family ((arxiv.org))
MeanCache, Gill et al.	Approximately 17% higher F-score and 20% higher precision on hit-or-miss decisions ((arxiv.org))	Relative to the baseline method it benchmarked against

Source	What was reported	Condition and limit
Anthropic prompt caching documentation	5-minute default lifetime ((platform.claude.com))	Prefix cache, not an answer cache. Different object entirely
OpenAI prompt caching documentation	30-minute eligibility window for GPT-5.6 and later models ((developers.openai.com))	Applies to GPT-5.6 and later; earlier models use in_memory or 24h retention. Prefix cache, not an answer cache

Table 3 shows a consistent pattern. Every figure that concerns answer correctness is conditional on a dataset, an embedding model or an attack model, and the only fully general statements are the documentation ones: similarity gives up a guarantee, and thresholds are tunable. The Scientific Reports figures are paper-reported rates for a specific simulated setting, not universal cache correctness. The authors say so in their own terms, noting that the results should not be interpreted as formal guarantees ((www.nature.com)). No cache-specific benchmark crossing question meaning with evidence state was found in the sources reviewed, which is the gap this report addresses.

Metric definitions

Let a **candidate** be a stored answer proposed for a request, and **valid** mean that both proof obligations hold for that candidate. For a fixture with raw counts:

- **False-reuse rate** = accepted invalid candidates divided by all accepted candidates. If no candidate is accepted, the rate is undefined, not zero and not perfect.
- **Invalid-candidate acceptance** = invalid candidates accepted divided by invalid candidates tested.
- **Valid-reuse recall** = valid candidates accepted divided by valid candidates tested.
- **Reuse answer rate** = cache-served eligible requests divided by all eligible requests.
- **Change exposure time** = last observed unsafe serve time minus declared change time, with sampling resolution and censored cases reported.
- **Latency and token savings** are measured and reported as secondary diagnostics, never as proxies for evidence correctness.

Reasons are reported separately: question mismatch, support invalidation and reader-scope failure. A failure can carry more than one reason, so reason counts must not be summed into unique failures.

A worked example with invented counts

These numbers are invented to show arithmetic and describe no system (Hypothetical Example). Suppose 40 candidates are tested, 24 valid and 16 invalid, drawn from 10 request families. A candidate semantic arm accepts 30: 22 valid and 8 invalid. Then false-reuse rate is **8/30 = 26.7%**, invalid-candidate acceptance is **8/16 = 50.0%**, and valid-reuse recall is **22/24 = 91.7%**. Reasons among the 8 invalid accepted: 5 question mismatch, 4 support invalidation and 1 reader scope, a total of 10 reason labels for 8 unique failures because two candidates failed on two grounds. Reporting 10 failures would overcount by two.

Uncertainty and clustering

Small fixtures need intervals. **Brown, Cai and DasGupta** recommend the Wilson interval or the equal-tailed Jeffreys prior interval for small samples (www.doi.org), and show that the erratic coverage of the standard Wald interval makes it a poor default (www.doi.org). For the example above, a Wilson 95% interval for 8/30 runs from about **14.2% to 44.4%**, which is wide, and that is before accounting for families. Paraphrases and neighbors of one seed answer are not independent observations. **Cameron and Miller** explain that when errors are correlated within clusters, default standard errors can greatly overstate estimator precision (cameron.econ.ucdavis.edu), define the clustering structure as errors uncorrelated across clusters but correlated within them (cameron.econ.ucdavis.edu), and note that a cluster bootstrap must resample entire clusters rather than individual observations (cameron.econ.ucdavis.edu). With few clusters, cluster-robust standard errors over-reject and intervals come out too narrow (cameron.econ.ucdavis.edu). With 10 request families, the honest report is the raw counts, a family-clustered interval and a statement that 10 clusters is a small number.

Implications and Future Directions

What this changes for teams operating answer caches

First, the acceptance question moves from "is the query close enough" to "which cell of the matrix is this request in." That makes a reuse decision reviewable by someone who was not present. Second, evidence staleness stops being a pure timing parameter: the FreshQA benchmark includes questions that require fast-changing world knowledge as well as questions with false premises ((arxiv.org)). The same logic applies to stored answers: some propositions change quickly and some do not, and the Category-aware caching paper's own assumption is that content such as stock prices, news and weather needs short lifetimes ((arxiv.org)), an assumption rather than a measurement.

Third, truth maintenance remains a data problem. Truth discovery research starts from the observation that multi-source information usually contains conflicts (arxiv.org) and resolves them by estimating the reliability of each source (arxiv.org). Knowledge Vault attaches calibrated probabilities to fused facts (research.google). A cached answer inherits the uncertainty of the evidence beneath it, and a reuse ledger that records source version and observation time keeps that uncertainty visible.

Where a semantic cache, a retrieval evaluation and a system of record each fit

A semantic cache is the right software category when the need is low-latency lookup of previously generated answers; it supplies candidate selection, and the matrix above tests it. A retrieval or evaluation framework answers different questions: the Ragas paper names faithfulness, the ability of the LLM to exploit retrieved passages in a faithful way ((arxiv.org)), and context relevance, the ability of retrieval to identify relevant and focused passages ((arxiv.org)). Neither is a substitute for the other. Reliable Context, the publisher's own product, is a system of record and is relevant only to the evidence side of the contract. It is not a semantic cache and does not replace one, and this report makes no claim about how it is built.

Open questions and limits

- **Public data:** the Scientific Reports evaluation data are available only on request, so no independent replication packet was established.
- **Provider semantics:** product thresholds, scoping and versioning change. LangCache semantics may change; verify them against the current Redis documentation before testing.
- **Needle-style tests:** Asana's engineering account of its own testing asks the model to find the correct data in a very large project ((asana.com)) and links to a reusable method from Arize, which embeds specific, targeted information in a larger body of text ((arize.com)). This shows only that concrete evaluation designs get reused when published in checkable form. It is not cache-specific evidence and transfers no performance figure.

Frequently Asked Questions (FAQs)

When is a cached answer still valid?

When both obligations hold: the new request is equivalent in proposition, scope and requested action to the one the answer was built for, and the evidence behind the answer is eligible for this request, reader and time. If either is unproven, the answer is a candidate, not a result.

How can a team know whether cached data is stale?

Elapsed time is only a proxy. HTTP caching distinguishes freshness from validation by an origin (www.rfc-editor.org), using validators of different strengths (www.rfc-editor.org). For answers, the equivalent is comparing the recorded source version and observation time with the current source state, and recording a reason when that state is unknown or blocked.

How does cache invalidation work for AI answers?

Three mechanisms are documented: expiration by time-to-live (redis.io), explicit removal when the underlying data changes (aws.amazon.com), and deletion by attribute (redis.io). Content delivery networks define invalidation as declaring cached content invalid (docs.cloud.google.com). None of these decide whether a paraphrased request is equivalent, which is why the matrix tests the question side separately.

What is the role of data provenance and lineage for decisions?

Provenance records where a value came from and the process by which it arrived (link.springer.com), and lineage standards such as OpenLineage record how datasets come into being (openlineage.io). For reuse decisions they supply the source, version, observation time and derivation fields.

How can a data-driven decision be defended in an audit?

By being able to reproduce what was known when the decision was made. Temporal data models distinguish valid time from transaction time (sigmodrecord.org), and system-versioned designs aim to keep the history of data changes from being tampered with (sigmodrecord.org). A reuse ledger plus preserved source versions

lets a reviewer reconstruct the answer, its evidence and the disposition taken. This is a technical record, not a statement of what any regulator accepts.

What is the risk of stale data in decision making?

Using an answer whose support has been corrected, retracted or superseded without noticing. The risk is not only elapsed time: Fowler notes that retroactive changes are a problem when actions are based on a past state ((martinfowler.com)).

What is a system of record for decision data?

A system of record is the place where a value has a single authoritative, cited form with an observation date. A semantic cache is a different thing: Redis calls its cache non-authoritative and rebuildable ((redis.io)).

What is a cache validity checklist?

- Is the new request equivalent in entity, date, polarity, unit and requested action?
- Is the source version unchanged, and is the observation time within the declared policy?
- Is the evidence unretracted, uncorrected, available and free of new exceptions?
- Is this reader eligible now?
- Is the disposition written to the ledger with a reason?

How can AI-generated answers be reused safely?

Select candidates by lookup, validate each against the matrix before serving, scope entries with hard metadata boundaries, and keep the ledger. Treat unknown support as a reason to re-evaluate, not as a pass.

Conclusion

A cached answer has two proof obligations that similarity does not discharge: it must answer this request, and its evidence must remain eligible for this request at this time. The paired reuse matrix makes both testable. It specifies the correct outcome for each combination of request meaning and evidence state before any system is run, separates false acceptance caused by question mismatch from false acceptance caused by evidence state, keeps candidate selection distinct from validity checking, and measures change exposure from declared observation points without promising instantaneous truth.

The evidence base is honest but thin. Provider documentation openly describes the correctness trade of similarity matching, one peer-reviewed study tests altered-meaning queries in a simulated setting, and several embedding studies show where similarity scores are weakest. No public benchmark that crosses request meaning with evidence state was found, and the cited study's data are available only on request. Teams should therefore treat the matrix as a fixture to build and publish, using synthetic or authorized data, and report raw counts with family-clustered intervals. A hit with unknown support is a candidate to be re-evaluated, and a served answer becomes a record only when its evidence, version and observation time are on file.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://redis.io/docs/latest/develop/ai/context-engine/langcache/concepts
2. https://redis.io/docs/latest/develop/use-cases/semantic-cache
3. https://www.nature.com/articles/s41598-026-36721-w
4. https://cameron.econ.ucdavis.edu/research/Cameron_Miller_JHR_2015_February.pdf
5. https://learn.microsoft.com/en-us/azure/api-management/llm-semantic-cache-lookup-policy
6. https://www.rfc-editor.org/rfc/rfc9111.html
7. https://platform.claude.com/docs/en/build-with-claude/prompt-caching
8. https://developers.openai.com/api/docs/guides/prompt-caching
9. https://docs.vllm.ai/en/latest/features/automatic_prefix_caching.html
10. https://www.rfc-editor.org/rfc/rfc7089.html
11. https://sigmodrecord.org/publications/sigmodRecord/1209/pdfs/07.industry.kulkarni.pdf
12. https://martinfowler.com/articles/bitemporal-history.html
13. https://arxiv.org/html/2502.03771v5
14. https://arxiv.org/abs/2510.26835
15. https://arxiv.org/abs/2305.07614
16. https://arxiv.org/abs/2306.05083
17. https://arxiv.org/abs/2509.05691
18. https://learn.microsoft.com/en-us/azure/api-management/llm-semantic-cache-store-policy
19. https://aws.amazon.com/blogs/database/optimize-llm-response-costs-and-latency-with-effective-caching
20. https://redis.io/docs/latest/operate/iris/langcache/create-service
21. https://www.rfc-editor.org/rfc/rfc9110.html
22. https://www.rfc-editor.org/rfc/rfc5861.html
23. https://redis.io/docs/latest/develop/ai/context-engine/langcache/api-examples
24. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html
25. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html
26. https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html
27. https://arxiv.org/abs/2602.13165
28. https://link.springer.com/chapter/10.1007/3-540-44503-X_20
29. https://www.w3.org/TR/prov-dm
30. https://www.w3.org/TR/prov-o
31. https://openlineage.io/docs/spec/object-model
32. https://arxiv.org/abs/2403.02694
33. https://www.itl.nist.gov/div898/handbook/apr/section1/apr131.htm
34. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf

35. https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf
36. https://www.doi.org/10.1214/SS/1009213286
37. https://arxiv.org/abs/2310.03214
38. https://arxiv.org/html/2510.26835v1
39. https://arxiv.org/abs/1505.02463
40. https://research.google/pubs/knowledge-vault-a-web-scale-approach-to-probabilistic-knowledge-fusion
41. https://arxiv.org/abs/2309.15217
42. https://asana.com/inside-asana/llm-testing-claude-sonnet-analysis
43. https://arize.com/blog-course/the-needle-in-a-haystack-test-evaluating-the-performance-of-llm-rag-systems
44. https://docs.cloud.google.com/cdn/docs/cache-invalidation-overview

THE PUBLISHER

About Reliable Context

Reliable Context is a system of record for organizations that fuse private and public information and must defend every value they act on.

The guarantees

One canonical record per real thing: duplicates resolve by rule, ambiguity is shown rather than hidden, and related organizations relate without ever being merged. A verbatim, cited source and an observation date sit behind every value. Unknowns are declared as not found, blocked or not applicable with a reason, never left as an empty cell. Any past state of the record is reproducible, together with what changed since.

Where it runs

It runs inside the customer's perimeter, one database per organization. Permitted sources, model routing, reader clearance and retention are set as the customer's own policies, and every value is exportable with its chain of custody for an auditor or regulator.

What it is not

It is not a vector database or a RAG framework, not a data catalog, MDM or data-quality suite, not a CRM or data-enrichment vendor, and not a chatbot or AI assistant. Where a reader needs one of those, the honest answer is to say so and point them there.

Visit

See Reliable Context.

Reliable Context is in production use in two sectors; more than six thousand of its identity decisions were made by rule and none has been reversed. Articles published here explain general practice, cite their primary and academic sources, and name the jurisdiction of any regulatory rule they describe. They are not legal, regulatory or compliance advice for a specific organization, and requirements change.

Website: https://reliablecontext.com

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.