When Semantic Caches Lie

Why One Similarity Threshold Cannot Make an LLM Cache Both Safe and Useful Across Domains

Sunny Patel*

Independent Researcher, Toronto, Canada

sunnypatel.net | sunnypatell/semantic-cache-reliability

Abstract

A semantic cache reuses a stored answer when a new prompt's embedding is close enough to an earlier one, trading a model call for a lookup. The property such a cache actually needs, whether one prompt's stored answer is valid for another, is not the cosine similarity it can measure, and the cache fails exactly where the two diverge. Grounding this *cache-equivalence* in human-labeled prompt pairs, we map the false-hit rate of six sentence embeddings across three domains with bootstrap confidence intervals. The sharpest finding is an inversion: on adversarial near-duplicates a classical lexical baseline separates equivalent from non-equivalent prompts better than every neural encoder, because the surface-form insensitivity that makes an embedding useful is what blinds it to a word-scrambled paraphrase; even a joint cross-encoder verifier does not recover the case, and a larger encoder barely helps. Holding the false-hit rate under one percent caps coverage near nine percent on news and under one percent elsewhere. Under a realistic penalty a fixed threshold costs several times more than not caching. A threshold calibrated to the measured cost recovers near-baseline cost, but transfers across domains only in the conservative direction. We release the protocol so that any team can measure this gap on its own encoder and price it into the threshold.

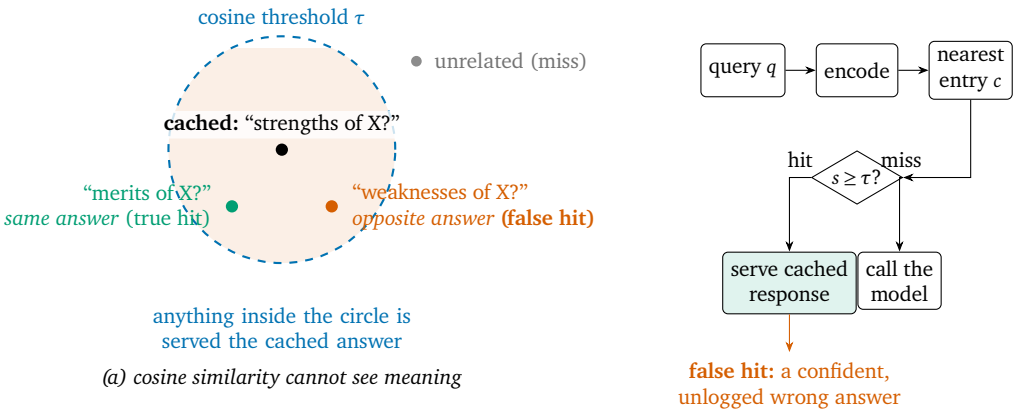
Keywords: semantic caching; large language models; embedding models; false-hit rate; retrieval-augmented generation; threshold calibration; reliability.

Plain-language summary: An LLM cache reuses an old answer when a new question looks similar. We show it often serves confident wrong answers when questions look alike but mean different things, measure the resulting damage, and show how to set the similarity cutoff that decides when reuse is safe.

1 Introduction

A semantic cache answers a new prompt with a stored response whenever the new prompt looks close enough to one the system has already served. Closeness is judged in an embedding space: the

 Correspondence: sunnypatel124555@gmail.com.



(a) cosine similarity cannot see meaning

(b) where the false hit is born

Figure 1 – **Cache-equivalence is not cosine similarity. (a)** A semantic cache serves the stored answer to any prompt whose embedding lies within a cosine threshold τ of a cached prompt (shaded). The genuine paraphrase "merits of X?" and the near-duplicate "weaknesses of X?" sit equally close to the cached "strengths of X?", yet only the first may reuse the stored answer; no threshold admits the true hit while rejecting the false one. **(b)** Because the cache fires on similarity alone, the false hit leaves as a confident, unlogged wrong answer. This paper measures how often this happens across encoders and domains, what it costs downstream, and how to price it into τ.

cache encodes the incoming query, finds its nearest stored neighbor, and returns the neighbor's answer when their cosine similarity clears a threshold τ. The appeal is obvious. A served hit skips a model call, so a cache that fires often turns the dominant cost of an LLM service, inference, into a lookup [1, 10]. Production systems now ship semantic caches by default, and recent work has pushed their hit rates higher and added online guarantees on correctness [14, 17].

The threshold hides a hard decision. Set it low and the cache fires often, paying back its keep, but it also begins to serve answers to prompts that only resemble an earlier one. Set it high and those mistakes vanish, along with most of the savings. The mistake itself is quiet. When the cache returns a stored answer to a prompt it does not actually fit, nothing errors and nothing logs; the user simply receives a confident, wrong reply. We call this a *false hit*, and it is the failure mode this paper measures.

The difficulty is that embedding similarity and answer reuse are not the same property (fig. 1): a semantic cache fails exactly where the two diverge. Two prompts can be near-identical in wording yet require opposite answers, as "what are the strengths of this design" and "what are the weaknesses of this design" do; two others can share almost no words yet take the same answer. Similarity is what the cache can see. *Cache-equivalence*, the question of whether one prompt's stored answer is valid for another, is what it needs. Prior work has largely optimized the first as a proxy for the second: it tunes thresholds, learns them online from correctness feedback [14], specializes them by query category [17], or sharpens the embedding itself [9]. What the literature has not done is measure how often the proxy fails when it is not under attack, chart how that failure rate moves with the embedding model and the domain, follow it into the quality of the task the cache serves, or turn the measurement into an operating point that accounts for what a false hit actually costs. The one adversarial study of these caches shows the proxy can be broken on purpose [21]; we ask how often it breaks on its own.

This paper answers that question with a measurement study and a calibration method built on it. We separate cache-equivalence from textual similarity and label it directly, drawing on human-annotated pair corpora whose judgments encode same-meaning rather than word overlap [4, 16, 20]. On this footing we measure the false-hit rate of six widely used sentence embeddings [13, 18, 19] across three domains and the full range of thresholds, report it with bootstrap confidence intervals, and read off the operating points a practitioner reasons about: how much of the traffic a cache can serve while holding its error below a ceiling, and how clean it can be while still reusing a target share of genuinely reusable answers. We then inject false hits into downstream retrieval and generation to measure what they cost once served, and we use that cost to calibrate the threshold. The resulting policy minimizes expected cost under the measured, asymmetric penalty of a false hit, and it transfers across models and domains without the per-prompt online feedback that earlier correctness-bounding methods require.

Our findings are not reassuring for the default of trusting a single similarity threshold. On adversarial paraphrases the strongest embedding reaches a ROC-AUC of only 0.67, and a lexical baseline outscores every neural encoder, because the surface-form insensitivity that makes an embedding useful is exactly what blinds it to a word-scrambled near-duplicate. Holding the false-hit rate under one percent caps coverage near nine percent on the easiest domain and under one percent on the others. Downstream, a false hit is nearly as costly when it corrupts the retrieved context as when it corrupts the final answer, because the reader rarely recovers from a wrong premise; what softens the damage is a milder corruption, not an earlier position in the pipeline. A fixed similarity threshold proves catastrophic under a realistic false-hit penalty, costing several times the price of not caching at all, while a threshold calibrated to the measured penalty recovers near-baseline cost and adapts to each domain's difficulty. Such a threshold transfers only in the conservative direction: a value tuned on an easy domain backfires when reused on a hard one.

Contributions.

- A *cache-equivalence* construct that separates "safe to reuse the stored answer" from "looks similar," together with a labeling protocol that grounds it in human judgments rather than embedding geometry (section 2).
- The first systematic map of benign false-hit economics for embedding caches: false-hit rate and coverage across six encoders, three domains, and all thresholds, with bootstrap confidence intervals and the operating points that matter in deployment (section 3).
- A measurement of the downstream cost of a false hit, obtained by controlled fault injection into retrieval and generation, broken down by task and by the cache's position in the pipeline (section 4).
- A cost-aware threshold-calibration policy that ties the operating point to the measured cost of a false hit. It exposes that a fixed default is catastrophic under a realistic penalty, matches a correctness-bounding baseline without a hand-set error target, and reveals that thresholds transfer across domains only in the conservative direction (section 5).

All code, the labeled data, and the exact commands that regenerate every number and figure are released with the paper.

Table 1 – Notation used throughout the paper.

Symbol	Meaning	Symbol	Meaning
q	incoming query (prompt)	$\textsc{fhr}(\tau)$	false-hit rate, 1 – precision
c	nearest stored entry	$\mathrm{cov}(\tau)$	coverage, fraction served
$\mathbf{e}$.	unit embedding of a prompt	c_{fh}	cost of a false hit (model calls)
s	cosine $\cos(\mathbf{e}_q, \mathbf{e}_c)$	$C(\tau)$	expected cost per query
τ	threshold; serve iff $s \geq \tau$	N	number of queries
TP, FP	true / false hits at τ	PR-AUC, ROC-AUC	threshold-free separability

2 What a cache must decide

2.1 The decision and its failure mode

A semantic cache holds a set of prompt–response pairs. When a query q arrives, the cache encodes it to a vector $\mathbf{e}_q$, retrieves the nearest stored entry c with vector $\mathbf{e}_c$, and computes the cosine similarity $s = \cos(\mathbf{e}_q, \mathbf{e}_c)$. If $s \geq \tau$ the cache serves c's stored response; otherwise the query falls through to the model. The single free parameter is the threshold τ.

A served query is a *hit*. A hit is *true* when the stored response is a valid answer to q and *false* when it is not. The quantity we track is the false-hit rate among served queries,

$$\textsc{fhr}(\tau) \;=\; \Pr[\text{ not equivalent} \mid s \geq \tau\,] \;=\; 1 - \text{precision}(\tau), \tag{1}$$

reported together with *coverage*, the fraction of queries the cache serves at τ. Coverage is the benefit and FHR is the silent cost, and the threshold trades one for the other. Table 1 collects the notation used throughout.

2.2 Cache-equivalence is not similarity

The label in eq. (1) is the crux. We define two prompts as *cache-equivalent* when the stored response to one is a factually and logically valid answer to the other, so that serving it requires no regeneration. This is a property of answers, not of surface form. It is distinct from cosine similarity, which is all the cache can observe, and the gap between the two is precisely where false hits live: a pair that scores high yet is not equivalent is a false hit waiting to happen.

Because cache-equivalence is a human judgment, we ground it in human-labeled data rather than in the embedding geometry we are auditing. We use three public pair corpora whose annotations encode same-meaning or same-intent (table 2). Quora Question Pairs labels whether two questions are duplicates, and duplicate questions are, by construction, questions that should receive the same answer [16]. The Microsoft Research Paraphrase Corpus labels sentential equivalence in a more formal, news register [4]. PAWS is the decisive case: its pairs are constructed to share almost all of their words while frequently differing in meaning, so it isolates the prompts that look identical to a cache yet must not share an answer [20]. Treating question-duplication and paraphrase as proxies for answer-equivalence is defensible but imperfect, and we return to the limits of the proxy in section 7.

Table 2 – The three labeled domains. Base rate is the fraction of pairs labeled cache-equivalent. PAWS pairs are built to maximize lexical overlap, which makes it the hardest case for a similarity cache.

Domain	Source	Pairs	Base rate
Questions (QQP)	Quora Question Pairs	2500	0.36
News (MRPC)	MS Research Paraphrase	2500	0.68
Adversarial (PAWS)	Word-scrambled paraphrase	2500	0.44

2.3 Encoders

We measure six sentence embedding models that span the sizes and families a practitioner would actually reach for: `all-MiniLM-L6-v2` and `all-mpnet-base-v2` from the Sentence-Transformers family [13, 15], the small and large E5 models [18], and the base and large BGE models [19]. Each model is used in its recommended configuration for symmetric similarity, and exact identifiers and revisions are recorded in the model cards (section A). A lexical TF-IDF cosine serves as a non-neural control, fit unsupervised on word unigrams and bigrams of the pair text with no use of the labels, which isolates how much of the cache's behavior comes from learned semantics rather than word overlap. We additionally evaluate a cross-encoder that scores each pair jointly rather than embedding the two prompts independently; it cannot serve as an indexable cache, but it bounds what a verification layer can achieve (section 3).

2.4 Metrics

Two prompts in a labeled pair give one similarity score and one equivalence label, so a domain–encoder combination yields a set of (label, s) observations. From these we report threshold-free separability first, because it does not depend on a chosen operating point: the area under the precision–recall curve (PR-AUC), which is informative under the class imbalance typical of cache traffic, and the area under the ROC curve (ROC-AUC) as a secondary summary. We then report the two operating points a deployment cares about. *Coverage at a false-hit ceiling* answers "if the cache may serve a wrong answer no more than once in a hundred, how much traffic can it serve?" *Precision at a recall target* answers its dual, "if the cache must reuse at least this share of genuinely reusable answers, how clean can it be?" Cosine scores are not comparable across encoders, so all cross-encoder comparison uses the threshold-free metrics, and per-encoder thresholds are calibrated separately (section 5).

2.5 Statistics

Every reported quantity carries a 95% bias-corrected and accelerated bootstrap confidence interval over pairs [6, 7], which suits the skewed sampling distribution of a rare-event rate better than a normal approximation. Differences between methods on a shared set use a paired bootstrap, and the many comparisons across encoders, domains, and thresholds are corrected for multiple testing with the Benjamini–Hochberg procedure [2]; we report effect sizes alongside every test [5]. The analysis plan, the primary metric, and the sample sizes were fixed before the main run in a plan released with the code (section E).

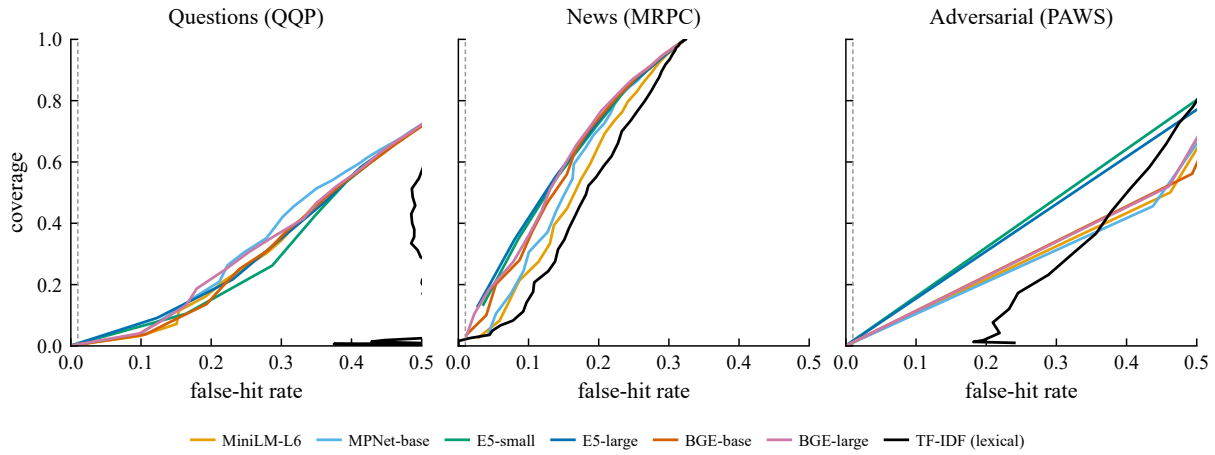


Figure 2 – The cost of safety. Each curve sweeps the similarity threshold for one encoder; the vertical axis is coverage (the fraction of queries served) and the horizontal axis is the false-hit rate among those served. Up-and-to-the-left is better. The dashed line marks a 1% false-hit ceiling. On questions and news, low-error operation costs most of the coverage; on the adversarial domain (right) the frontier collapses toward the diagonal, where coverage can be bought only by accepting an almost equal rate of wrong answers. Bootstrap intervals on the headline operating point are shown in fig. 5.

3 How often the proxy fails

We report separability first, because it does not depend on a chosen threshold, then the false-hit/coverage trade-off (fig. 2), the score distributions that explain it, and the operating points a deployment would actually run.

3.1 Separability across models and domains

Figure 3 and table 3 summarize threshold-free separability as PR-AUC for every encoder–domain pair. Three patterns hold. First, the domain matters more than the model. For a fixed encoder the spread of PR-AUC across domains is wide, 0.62 to 0.90 for E5-large, while the spread across the six neural encoders within a fixed domain is narrow, 0.54 to 0.62 on the adversarial set. Second, the adversarial domain is close to unsolved: the best encoder reaches PR-AUC = 0.62 on PAWS against a base rate of 0.44, and its ROC-AUC of 0.67 is barely above the 0.5 of chance. Scale buys little: moving from E5-small to E5-large raises PAWS PR-AUC by 0.08, and the BGE base-to-large gain is 0.04. Third, and most telling, the lexical control inverts the order. TF-IDF trails the neural encoders badly on questions, 0.49 against 0.74 to 0.78, yet on the adversarial domain it *outscores every neural encoder*, 0.66 against at most 0.62. The inversion is not a sampling artifact. A paired bootstrap over the shared pairs puts TF-IDF's PR-AUC above every neural encoder by 0.04 to 0.12 and its ROC-AUC by 0.06 to 0.13, and each of the twelve gaps survives a Benjamini–Hochberg correction across the comparison family (largest corrected $p < 0.003$). Word-scrambling defeats the semantics the neural models encode, while the bigram structure the lexical model reads survives it. The very property that makes an embedding useful, its insensitivity to surface form, is also what blinds it to an adversarial paraphrase.

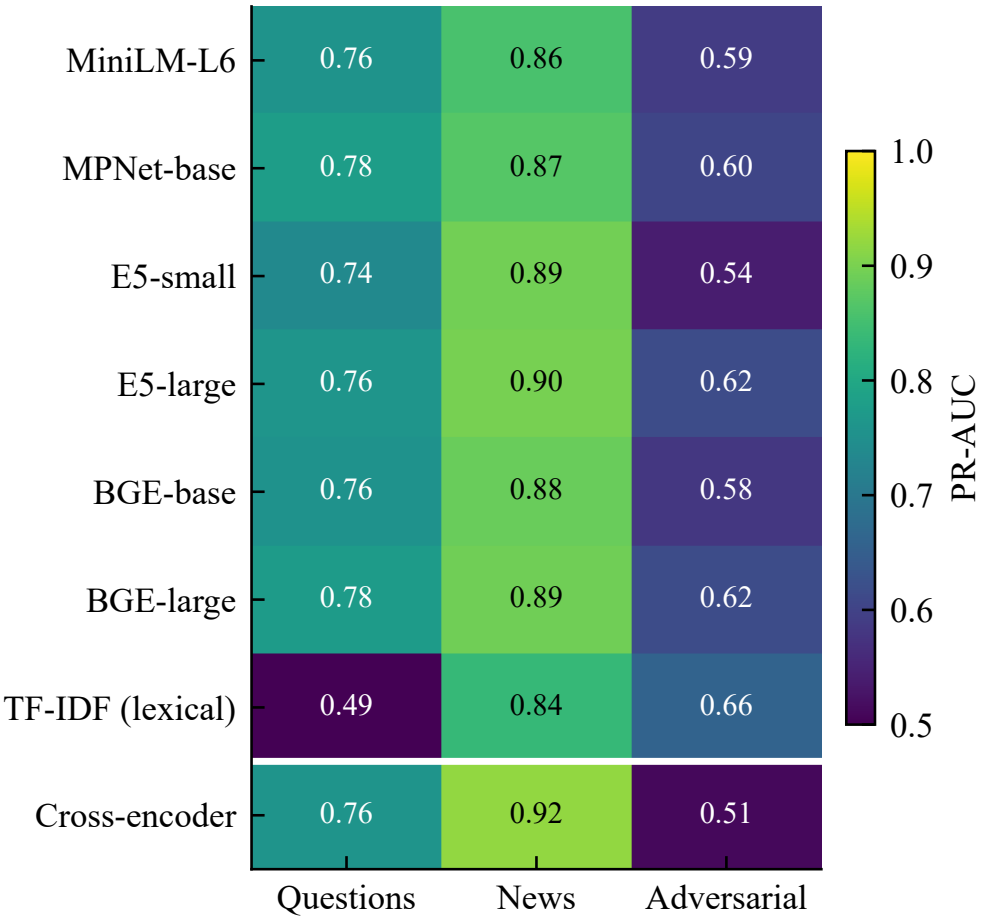


Figure 3 – Threshold-free separability (PR-AUC) for each encoder and domain. Higher is better; the base rate of equivalent pairs is 0.36 (questions), 0.68 (news), and 0.44 (adversarial). The adversarial column stays near the floor for every encoder.

A cross-encoder verifier does not rescue the hard case. A bi-encoder is handicapped by construction: it commits each prompt to a vector before it sees the other, so it cannot react to the specific pair in front of it, which is the blindness the adversarial domain exploits. A cross-encoder reads both prompts jointly and could, in principle, catch a word-scrambled near-duplicate, so it is the obvious fix and the natural *verification* layer the rest of this paper recommends. We measure an off-the-shelf semantic-similarity cross-encoder (`stsb-roberta-base`; section A) on the same pairs; it cannot itself be the cache, because it scores each pair on the fly and cannot be indexed, but it sets the ceiling a verifier could reach. It is the single strongest method on news (PR-AUC 0.92) and matches the bi-encoders on questions, yet on the adversarial domain it collapses to PR-AUC 0.51, below the lexical control and barely above the 0.44 base rate (table 3). Joint attention helps where meaning and surface form already agree; it does not recover the case where they diverge. The difficulty of the adversarial domain is a property of the equivalence judgment, not an artifact of the bi-encoder architecture, and a verification layer inherits it.

Table 3 – Threshold-free separability for every method and domain (point estimates; bold is the column best). Higher is better. The neural bi-encoders cluster near the adversarial base rate of 0.44; the lexical control and even a joint cross-encoder do not rescue that column, and the lexical control is the single best method on it.

	PR-AUC ↑			ROC-AUC ↑		
Encoder	Ques.	News	Adv.	Ques.	News	Adv.
MiniLM-L6	0.76	0.86	0.59	0.87	0.76	0.64
MPNet-base	**0.78**	0.87	0.60	**0.89**	0.78	0.66
E5-small	0.74	0.89	0.54	0.86	0.81	0.59
E5-large	0.76	0.90	0.62	0.87	0.82	0.67
BGE-base	0.76	0.88	0.58	0.87	0.80	0.63
BGE-large	**0.78**	0.89	0.62	0.88	0.81	0.65
TF-IDF (lexical)	0.49	0.84	**0.66**	0.70	0.71	**0.72**
Cross-encoder[a]	0.76	**0.92**	0.51	0.87	**0.84**	0.60

Domains: Questions (QQP, base rate 0.36), News (MRPC, 0.68), Adversarial (PAWS, 0.44). 95% BCA bootstrap half-widths are ≤ 0.04 throughout; full intervals and operating points are in table 6.

[a] A joint cross-encoder verifier, not a drop-in cache: it scores each pair on the fly and cannot be ANN-indexed. Shown for reference.

3.2 The false-hit/coverage frontier

Figure 2 reads the same data as an operating curve. To hold the false-hit rate under 1%, a deployment must move far up the threshold, and coverage falls with it. On news the best encoder (E5-large) serves about 9% of traffic at a 1% ceiling; on questions it serves under 1%; on the adversarial domain it serves essentially nothing. Loosening the ceiling to 5% raises news coverage to about 24% but leaves questions under 3% and the adversarial domain under 1%. The news frontier is the most forgiving because its pairs are the least lexically confusable; the adversarial frontier is the cautionary case, because it is the one a real cache meets whenever two prompts differ in a small but decisive word.

3.3 Why: the grey zone

Figure 4 shows where the failure comes from. For BGE-large, the similarity scores of equivalent and non-equivalent pairs are well separated on questions but pile into the same narrow, high band on the adversarial domain. On PAWS the median similarity of equivalent and non-equivalent pairs differs by only 0.011, both above 0.97, against a gap of 0.19 on questions; no threshold can divide distributions that overlap this tightly. A cache cannot act on a distinction its encoder does not represent, and on adversarial near-duplicates the encoder does not represent it.

3.4 Operating points

Figure 5 states the headline as a single number per encoder and domain: the coverage a cache can deliver while keeping its false-hit rate at or below 1%. Across encoders the achievable coverage is under 1% on questions, between roughly 0.5% and 9% on news, and below 0.5% on the adversarial domain; table 6 gives the full operating-point matrix behind these numbers. A single

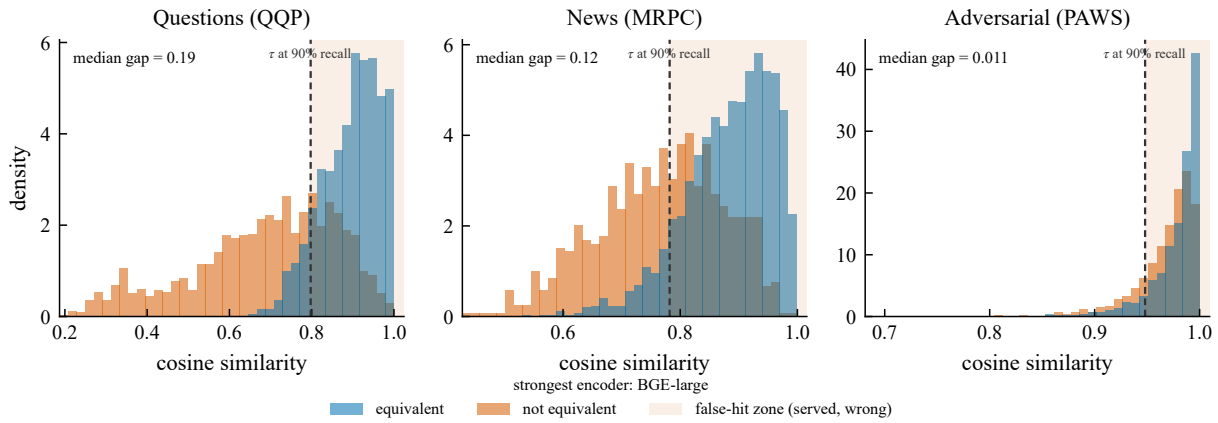


Figure 4 – The grey zone. Similarity-score distributions for equivalent (blue) and non-equivalent (orange) pairs, for the strongest encoder. The dashed line is the threshold that retains 90% of equivalent pairs; the shaded band to its right is the false-hit zone, the non-equivalent mass a cache at that threshold would serve. Clean separation on questions (median gap 0.19) becomes near-total overlap on the adversarial domain (median gap 0.011, both classes above 0.97), where the false-hit zone swallows most of the non-equivalent pairs.

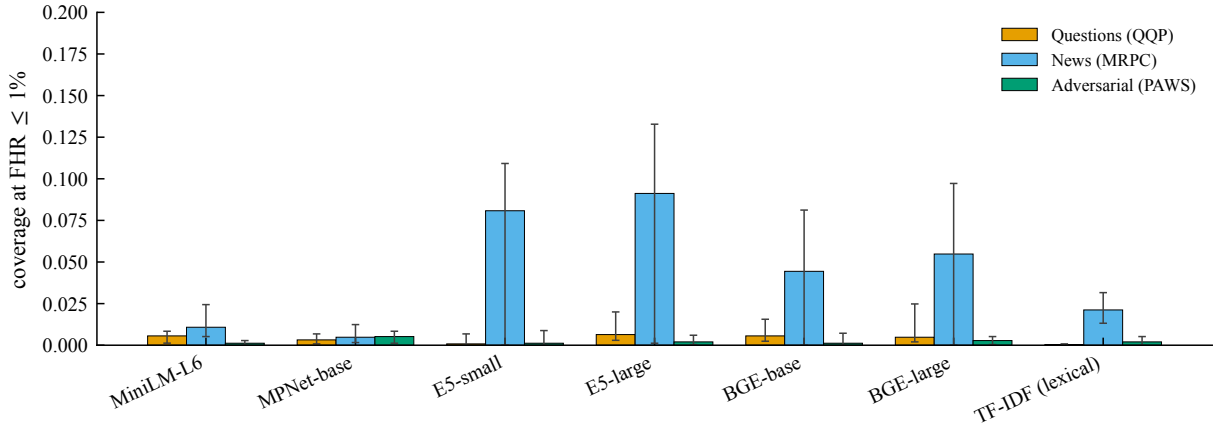


Figure 5 – Coverage achievable at a 1% false-hit ceiling, per encoder and domain. Error bars are 95% bootstrap intervals. Even the strongest encoder serves only a small share of questions and news under the ceiling, and almost nothing on the adversarial domain.

global threshold chosen for a clean domain will leak false hits in a harder one, and the adversarial domain admits no operating point that is both safe and useful.

4 What a false hit costs downstream

A false-hit rate matters only as much as the damage a false hit does once it is served, and that damage depends both on where the cache sits and on how badly it corrupts what the system sees. A cache on the final answer returns a wrong answer outright. A cache earlier in the pipeline, on the retrieved context of a retrieval-augmented generator, hands the reader a wrong premise; whether the reader recovers is an empirical question, and the answer turns out to be mostly no.

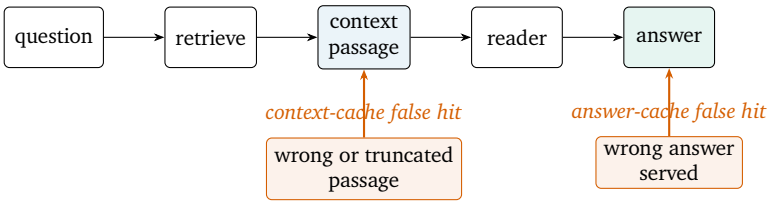


Figure 6 – **Where a served false hit corrupts the task.** In a retrieval-augmented question-answering pipeline a false hit can replace the final answer (the answer-cache position) or the passage the reader sees (the context-cache position). We inject at both, and at a milder truncation of the passage, and measure the end-to-end cost (fig. 7).

Protocol. We inject false hits into an extractive question-answering reader and measure end-to-end token F_1 against the reference answers. For each test question the reader runs on its correct context and on three faulted inputs that place a false hit at different points (fig. 6): a *wrong answer* served directly (the answer-cache position, where a different question's stored answer is returned verbatim), a *wrong passage* fed to the reader (the context-cache position, where a different question's passage is retrieved), and a *truncated passage* (a milder corruption of the same position). Because each question is faulted independently with probability p and carries at most one injected fault, the expected end-to-end F_1 at injected false-hit rate p is, by linearity of expectation, the exact mixture $(1-p)\,F_1^{\text{correct}} + p\,F_1^{\text{fault}}$. We compute it from the two measured endpoints over 400 questions with a fixed reader and report it for $p \in \{1, 2, 5, 10\}$%; the form is an accounting identity over independent injections, not an assumption about errors compounding across the pipeline.

Findings. The reader scores $F_1 = 0.79$ on correct context. Figure 7 shows what a false hit costs. Position does not protect the answer: a wrong served answer drops F_1 by 0.79 per unit of false-hit rate, and a wrong retrieved passage drops it by almost as much, 0.78, because the reader almost never extracts the right answer from a wholly wrong passage. What changes the cost is the *severity* of the corruption, not the position. A truncated passage costs only 0.48 per unit, because the answer survives the truncation often enough for the reader to find it. Concretely, at a 10% false-hit rate end-to-end F_1 falls from 0.79 to 0.71 for a wrong answer or a wrong passage, and to 0.74 for a truncated one. The lesson for calibration is plain: a false hit is expensive wherever it lands, so the penalty the next section trades against is large, and the common hope that a downstream reader will quietly absorb a wrong cached premise is not supported here.

5 Calibrating the threshold to cost

The frontier in section 3 shows the trade-off but does not choose where on it to operate. That choice is the threshold-setting problem, and it depends on how much a false hit costs relative to a miss. A miss recomputes an answer the cache could have reused, costing one model call. A false hit serves a wrong answer, costing the downstream damage measured in section 4 plus, often, the eventual recomputation. The two are not symmetric, and the asymmetry is the whole decision.

Policies. We compare three ways of choosing the per-encoder threshold, each fit on a calibration split and evaluated on a held-out test split of the same domain. The *fixed* policy uses a single hand-

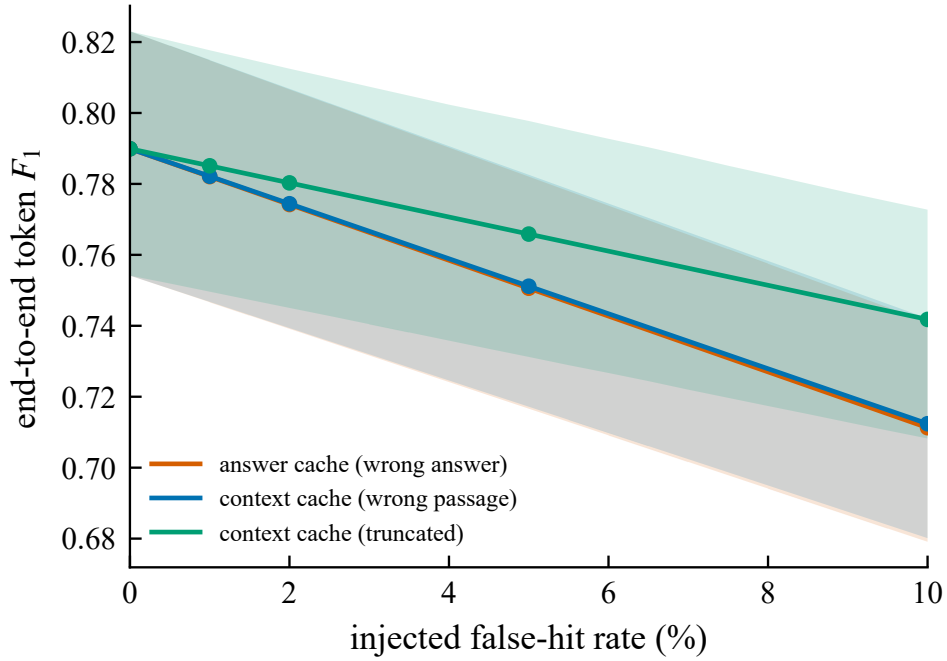


Figure 7 – End-to-end reader F_1 as the injected false-hit rate rises, for a false hit on the final answer and on the retrieved passage (wrong or truncated). A wrong answer and a wrong passage are almost equally damaging; only a partial corruption is gentler. Bands are 95% bootstrap intervals over the 400 questions; because all three conditions use the same questions, the separation between the truncated curve and the other two is a paired difference, tighter than the absolute bands suggest.

set similarity value, the common default. The *ceiling* policy chooses the lowest threshold whose calibration false-hit rate stays under a target, the strategy of correctness-bounding caches such as verified caching [14]. Our *cost-aware* policy chooses the threshold that minimizes expected cost,

$$C(\tau) \;=\; \frac{c_{\mathrm{fh}}\,\mathrm{FP}(\tau) + \big(\mathrm{TN}(\tau) + \mathrm{FN}(\tau)\big)}{N}, \tag{2}$$

where a correct hit saves a call, a miss costs one, and a false hit costs c_{fh} calls. The penalty is a deployment choice that folds the eventual recompute together with the reputational and task cost of a wrong answer (section 4 shows it is large). It is the only free input, so the policy needs no per-prompt online feedback (algorithm 1). We report a plausible bracket, $c_{\mathrm{fh}} \in \{5, 20\}$; the conclusions below are not an artifact of one penalty, because a larger penalty only pushes the chosen threshold higher, as it should.

Results. The comparison in table 4 and fig. 8 is stark. At a false-hit penalty of $c_{\mathrm{fh}} = 20$, the fixed $\tau = 0.80$ default costs 6.3 model calls per query against the always-recompute baseline of 1.0, because at that threshold the cache fires on most queries while leaking a 39% false-hit rate. A cache run on a fixed default is not a small loss; it is several times more expensive than not caching at all. Both principled policies avoid the disaster by retreating to a near-saturating threshold, and both land at almost exactly the baseline cost (0.99). The picture is not an artifact of the high penalty: even at the milder $c_{\mathrm{fh}} = 5$ the fixed default still loses, at 1.8× the no-cache baseline, while the principled policies stay at or below it. The cache earns a genuine saving only

Algorithm 1 – Cost-aware threshold calibration.

Require: calibration pairs $\{(s_i, y_i)\}_{i=1}^{N}$, with $y_i = 1$ iff the pair is cache-equivalent; false-hit penalty c_{fh}
Ensure: threshold $\tau^\star$ that minimizes expected cost $C(\tau)$

$C^\star \leftarrow \infty$; $\tau^\star \leftarrow 1$
for all candidate τ in the sorted unique scores $\{s_i\}$ **do**
  FP $\leftarrow |\{i : s_i \geq \tau,\ y_i = 0\}|$ {false hits served}
  $M \leftarrow |\{i : s_i < \tau\}|$ {misses, recomputed}
  $C(\tau) \leftarrow (c_{\text{fh}} \cdot \text{FP} + M)/N$ {a true hit saves the call}
  if $C(\tau) < C^\star$ **then**
    $C^\star \leftarrow C(\tau)$; $\tau^\star \leftarrow \tau$
  end if
end for
return $\tau^\star$

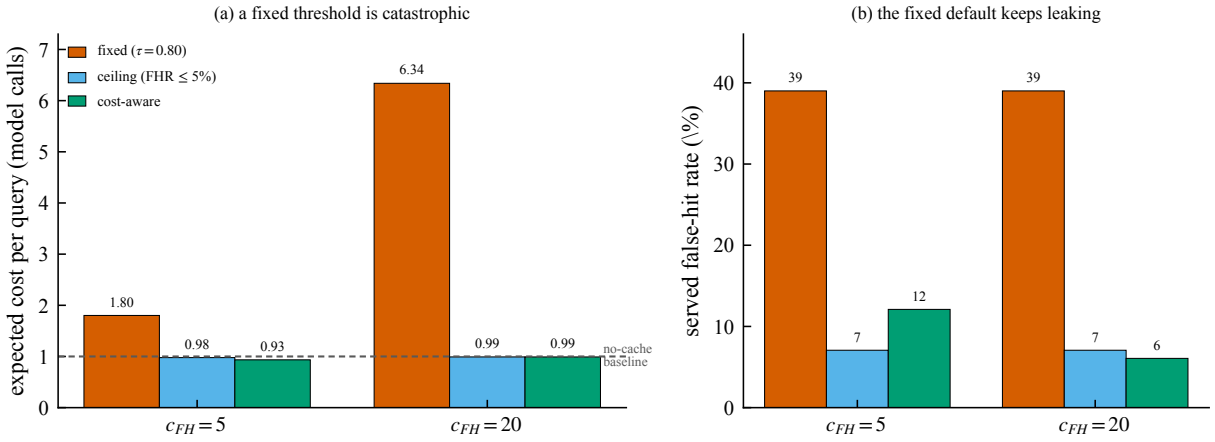


Figure 8 – **A fixed threshold is catastrophic under a realistic penalty; a calibrated one is not. (a)** Expected cost per query, in model calls against the always-recompute baseline of 1, for the three policies at two false-hit penalties, averaged over encoders and domains (the numbers behind table 4). The fixed $\tau = 0.80$ default costs several times the baseline and worsens as the penalty rises, while the ceiling and cost-aware policies sit on it. **(b)** Why: the fixed threshold serves the same 39% false-hit rate whatever the penalty, because its operating point never moves, while the principled policies hold the rate far lower, the cost-aware one tightening as the penalty grows.

where the domain allows it: on news the cost-aware policy holds the false-hit rate near zero while still serving enough to cut expected cost by about 7% for E5-large; on questions it roughly breaks even; on the adversarial domain it correctly serves almost nothing, because any coverage there costs more in false hits than it saves. The cost-aware and ceiling policies track each other closely, with neither strictly dominating; the value of the cost-aware form is that it derives the operating point from the measured cost ratio rather than from a hand-set error target.

Transfer. A threshold is only useful if it survives a change of domain, and here it does so in only one direction (fig. 10). A conservative threshold calibrated on the adversarial domain, which sits near 1.0, transfers to the easier domains at a cost gap no greater than 0.072 per query, and under

Table 4 – Threshold policies on held-out test data, averaged over encoders and domains at a false-hit penalty of $c_{\text{fh}} = 20$. Expected cost is in model calls per query against the always-recompute baseline of 1; lower is better (bold). A fixed default is several times more expensive than not caching at all; the principled policies recover the baseline, and the cost-aware form reaches it without a hand-set error target.

Policy	FHR (%)	Coverage (%)	Expected cost ↓
Fixed ($\tau = 0.80$)	39.0	71.1	6.339
Ceiling (FHR $\leq 5\%$)	7.1	2.6	0.991
Cost-aware (algorithm 1)	6.1	1.9	**0.991**

At the milder $c_{\text{fh}} = 5$ the fixed default still loses (1.8× baseline) while both principled policies stay at or below it; see section 5.

0.02 for most encoders. The reverse is ruinous: the 0.97 threshold that is right for news, applied to adversarial traffic, leaks so many false hits that expected cost jumps to 7.1, a gap of more than six calls per query. A threshold learned where the cache works cannot be trusted where it does not. The recommendation is concrete: calibrate per domain, and when in doubt err toward the conservative threshold, because the cost of an over-tight cache is a few missed reuses while the cost of an over-loose one is unbounded.

6 Related work

Semantic caching for language models. Systems that reuse model outputs by embedding similarity have moved quickly from prototype to production. GPTCache established the pattern of encoding a query, retrieving its nearest stored neighbor, and serving the neighbor's response above a similarity threshold [1]. Later work improved the hit rate and the threshold itself: MeanCache personalizes the decision with federated learning [10], category-aware caching sets different thresholds for different query types [17], and domain-specific embeddings sharpen the similarity signal with synthetic fine-tuning data [9]. The work closest to ours is vCache, which learns a per-prompt threshold online from correctness feedback and bounds the error rate with a statistical guarantee [14]. These systems share an objective: maximize reuse subject to keeping answers correct, where correctness is judged on the served queries.

What has not been measured. The literature reports hit rates, latency, and, in the case of verified caching, a bound on error [10, 14]. It does not chart how the benign false-hit rate moves across embedding models and domains, it does not follow a false hit into the quality of the downstream task, and it does not turn the cost of a false hit into an operating point. Our contribution is orthogonal to building a better cache: we measure the failure that every similarity cache shares and calibrate against its measured cost. The one study that probes these caches adversarially shows that an attacker can craft collisions that force false hits [21]; we ask the complementary question of how often the same failure occurs without an adversary.

Embeddings and their evaluation. The encoders we audit are standard sentence embedding models trained with contrastive objectives: Sentence-BERT and its MPNet backbone [13, 15],

the weakly supervised E5 models [18], and the BGE models from the C-Pack release [19], all of which rank highly on the Massive Text Embedding Benchmark [12]. Our labeled domains come from human-annotated paraphrase and duplicate-question corpora [4, 16, 20] rather than from semantic-similarity regression [3], because the question a cache must answer is binary equivalence, not graded similarity.

Measurement methodology. We follow the empirical-rigor practices that the evaluation literature has converged on: confidence intervals from the bias-corrected and accelerated bootstrap [6, 7], multiple-comparison correction by controlling the false discovery rate [2], and significance testing chosen for the structure of the data rather than by default [5]. We document the data and the models with a datasheet and model cards in the spirit of [8, 11].

7 Threats to validity

Construct validity. The central risk is the one we have been explicit about: our labels encode paraphrase and question-duplication, and we read them as cache-equivalence. The two come apart at the edges, and the gap runs both ways: a labeled non-paraphrase may occasionally admit the same answer, which inflates the measured false-hit rate, while a labeled paraphrase may need a different answer in context, which deflates it. We accept this gap deliberately, because the alternative, bespoke answer-equivalence annotation at scale, would trade a transparent, reproducible label for a private one. We therefore do not claim the residual bias favors our thesis. The claim is narrower: the labels track a public, reproducible notion of sameness, and the adversarial domain is built to isolate exactly the case a similarity cache cannot resolve, near-identical wording with divergent meaning. The numbers should be read as the reliability of the similarity proxy against that human notion of sameness, which is the proxy a deployed cache actually relies on.

Internal validity. Cosine scores are not comparable across encoders, so every between-encoder statement uses threshold-free metrics (PR-AUC, ROC-AUC), and every threshold is calibrated per encoder (section 5). Within an encoder, the false-hit rate and coverage are exact functions of the score distributions, so there is no estimator variance beyond sampling, which the bootstrap intervals capture.

External validity. We measure three domains and six encoders, and the specific numbers will not transfer verbatim to a new corpus or a next-generation embedding. The contribution that does transfer is the method: define cache-equivalence, collect a labeled pair set for the target domain, read the false-hit/coverage frontier, and calibrate the threshold to the measured cost. We designed the study so that re-running it on a new encoder is a single command. The adversarial domain is intentionally a worst case rather than an average one; real traffic will sit between the news and adversarial frontiers, and where it sits is itself something a deployment can measure with this protocol. Coverage is reported over balanced labeled pairs, so the absolute hit rate a live system sees will depend on how much of its traffic is genuinely reusable; the transferable object is the shape of the false-hit/coverage trade-off at a given ceiling, not the single coverage number.

Conclusion validity. All intervals are 95% bias-corrected and accelerated bootstrap intervals over pairs. Cross-encoder comparisons use a paired bootstrap over the shared pairs and are cor-

rected for the false discovery rate across the comparison family with the Benjamini–Hochberg procedure; the lexical-versus-neural gaps on the adversarial domain, for instance, hold at a corrected $p \leq 0.003$. We report the size of each difference next to its test, so that a reliable difference is not mistaken for a large one. The analysis plan, the primary metric, and the sample sizes were fixed before the main run and are released with the code.

8 Limitations

The study is offline and pairwise. It measures the quality of the similarity decision, not the dynamics of a live cache, so cache size, eviction, and the time-evolution of the stored set are out of scope; the false-hit rate we report is a property of the decision rule that any such system inherits. We do not evaluate exact-match or two-tier caches as separability baselines, because our pairs are non-identical by construction and an exact-match layer never fires on them; a two-tier system simply adds our semantic layer on top of a free exact layer, so our frontier is the part that governs its risk. The downstream cost in section 4 comes from controlled fault injection rather than from a production incident log, and we report it under several injection models to show the conclusion does not hinge on one. The wrong-passage condition injects an unrelated passage, which is harsher than the near-duplicate a real false hit would serve, so its cost is best read as an upper bound; the truncated-passage condition brackets the gentler end, and a real false hit sits between them.

9 Broader impact

A false hit is a silent error: the system returns a confident answer that no log flags as wrong. In low-stakes settings this trades a little quality for a large cost saving, which is the point of caching. In settings where a wrong answer carries real cost, such as health, legal, or financial guidance, the same mechanism can serve stale or incorrect information without any signal to the user. Our results argue that such systems should not run a single global similarity threshold; they should calibrate per domain to a false-hit ceiling tied to the cost of being wrong, and they should monitor the served stream rather than trust it. Embedding caches also have a documented adversarial surface: an attacker who can shape prompts can manufacture collisions that force false hits [21]. Our study quantifies the benign version of the same failure, and the defense is aligned in both cases: a conservative, cost-aware threshold, and, where the latency budget allows, a verification step rather than blind trust in similarity. Our cross-encoder measurement tempers the second lever: a joint verifier sharpens the easy and middling cases but does not recover adversarial near-duplicates (section 3), so on the hardest prompts the dependable choice is to decline to cache. On the other side of the ledger, caching reduces the compute and energy of repeated inference, and a more reliable cache captures that saving without silently degrading the answers it serves.

10 Conclusion

A semantic cache bets that two prompts which look alike can share an answer. We measured how often that bet loses when no one is attacking it. Separating cache-equivalence from similarity and labeling it with human judgments, we mapped the false-hit rate of six embedding models across three domains, traced a false hit into the task it corrupts, and calibrated the threshold

to the cost of being wrong. The picture is sober. On clean domains a cache can be kept honest, but only by surrendering most of its coverage; on adversarial near-duplicates the similarity signal carries almost no information about equivalence, and no threshold makes the cache both safe and useful. The remedy is not a better single number but a deliberate one: calibrate per domain against a false-hit ceiling set by what a wrong answer costs, and verify rather than trust. We release the labeled data, the harness, and the exact commands so that any team can run the same measurement on its own encoder and its own traffic before it decides what its cache is allowed to guess.

Data and code availability

All code, the labeled cache-equivalence dataset, and the exact commands that regenerate every table and figure are available at github.com/sunnypatell/semantic-cache-reliability and are archived under a DOI minted on release. The dataset is derived from the public QQP, MRPC, and PAWS corpora, which retain their own licenses.

Declarations

Funding. This research received no external funding.
Competing interests. The author declares no competing interests.
Acknowledgments. The author thanks the maintainers of the open corpora and embedding models this study builds on.

References

[1] F. Bang. “GPTCache: An Open-Source Semantic Cache for LLM Applications Enabling Faster Answers and Cost Savings.” In: *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS)*. 2023. DOI: 10.18653/v1/2023.nlposs-1.24.

[2] Y. Benjamini and Y. Hochberg. “Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing.” In: *Journal of the Royal Statistical Society: Series B (Methodological)* 57.1 (1995), pp. 289–300. DOI: 10.1111/j.2517-6161.1995.tb02031.x.

[3] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. “SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation.” In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017.

[4] B. Dolan and C. Brockett. “Automatically Constructing a Corpus of Sentential Paraphrases.” In: *Proceedings of the 3rd International Workshop on Paraphrasing (IWP2005)*. 2005, pp. 9–16.

[5] R. Dror, G. Baumer, S. Shlomov, and R. Reichart. “The Hitchhiker’s Guide to Testing Statistical Significance in Natural Language Processing.” In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 1383–1392. DOI: 10.18653/v1/P18-1128.

[6] B. Efron. “Better Bootstrap Confidence Intervals.” In: *Journal of the American Statistical Association* 82.397 (1987), pp. 171–185. DOI: 10.1080/01621459.1987.10478410.

[7] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[8] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III, and K. Crawford. "Datasheets for Datasets." In: *Communications of the ACM* 64.12 (2021), pp. 86–92. DOI: 10.1145/3458723.

[9] W. Gill, J. Cechmanek, T. Hutcherson, S. Rajamohan, J. Agarwal, M. A. Gulzar, M. Singh, and B. Dion. "Advancing Semantic Caching for LLMs with Domain-Specific Embeddings and Synthetic Data." In: *arXiv preprint* (2025). arXiv: 2504.02268.

[10] W. Gill, M. Elidrisi, P. Kalapatapu, A. Ahmed, A. Anwar, and M. A. Gulzar. "MeanCache: User-Centric Semantic Caching for LLM Web Services." In: *arXiv preprint* (2024). arXiv: 2403.02694.

[11] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. "Model Cards for Model Reporting." In: *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*. 2019. DOI: 10.1145/3287560.3287596.

[12] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. "MTEB: Massive Text Embedding Benchmark." In: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 2023.

[13] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2019. DOI: 10.18653/v1/D19-1410.

[14] L. G. Schroeder, A. Desai, A. Cuadron, K. Chu, S. Liu, M. Zhao, S. Krusche, A. Kemper, M. Zaharia, and J. E. Gonzalez. "vCache: Verified Semantic Prompt Caching." In: *arXiv preprint* (2025). arXiv: 2502.03771 [cs.LG].

[15] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. "MPNet: Masked and Permuted Pre-training for Language Understanding." In: *arXiv preprint* (2020). arXiv: 2004.09297.

[16] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding." In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*. 2019.

[17] C. Wang, X. Liu, Y. Zhu, A. Youssef, P. Nagpurkar, and H. Chen. "Category-Aware Semantic Caching for Heterogeneous LLM Workloads." In: *arXiv preprint* (2025). arXiv: 2510.26835.

[18] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei. "Text Embeddings by Weakly-Supervised Contrastive Pre-training." In: *arXiv preprint* (2022). arXiv: 2212.03533.

[19] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie. "C-Pack: Packed Resources For General Chinese Embeddings." In: *arXiv preprint* (2023). arXiv: 2309.07597.

[20] Y. Zhang, J. Baldridge, and L. He. "PAWS: Paraphrase Adversaries from Word Scrambling." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*. 2019.

[21] Z. Zhang, Z. Liu, Y. Xie, Q. Huang, and D. She. "From Similarity to Vulnerability: Key Collision Attack on LLM Semantic Caching." In: *arXiv preprint* (2026). arXiv: 2601.23088.

Table 5 – The six audited encoders. Dimensions and parameter counts are the published values for each checkpoint.

Model	Params (M)	Dim	License	Hugging Face identifier
MiniLM-L6	22.7	384	Apache-2.0	`sentence-transformers/all-MiniLM-L6-v2`
MPNet-base	109.0	768	Apache-2.0	`sentence-transformers/all-mpnet-base-v2`
E5-small	33.0	384	MIT	`intfloat/e5-small-v2`
E5-large	335.0	1024	MIT	`intfloat/e5-large-v2`
BGE-base	109.0	768	MIT	`BAAI/bge-base-en-v1.5`
BGE-large	335.0	1024	MIT	`BAAI/bge-large-en-v1.5`

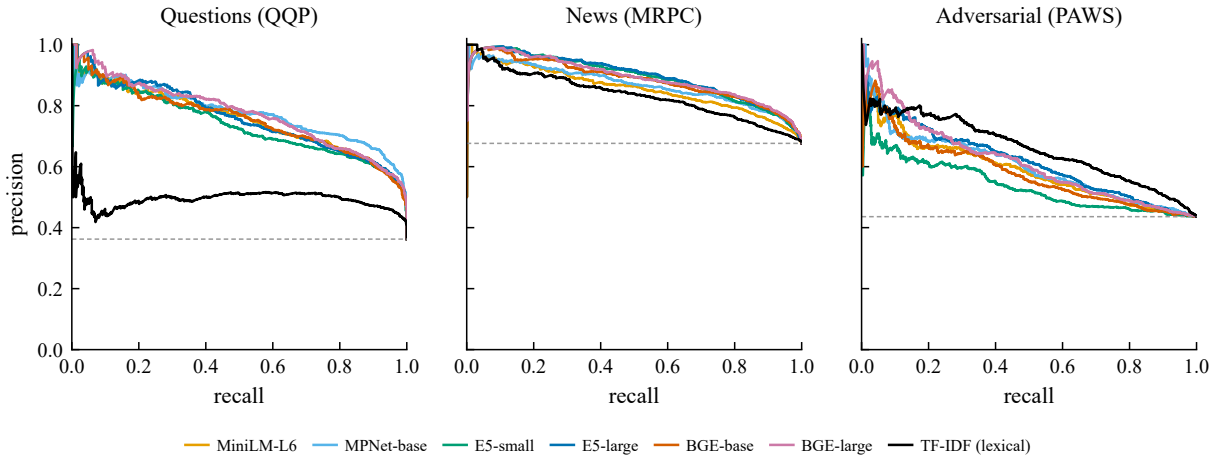


Figure 9 – Precision–recall curves behind the threshold-free separability of table 3. Each curve is one encoder and the dashed line is the domain base rate. On the adversarial domain (right) every neural encoder hugs the base rate while the lexical control rides above it, the same inversion the main text reports.

A Model cards

Table 5 records the six encoders. Each is used through the Sentence-Transformers interface with mean pooling and L_2-normalized output. The two E5 models are queried with the recommended `query:` prefix for symmetric similarity; the others take the raw text. Exact weight revisions are pinned in the released code so that the numbers reproduce against fixed checkpoints. The cross-encoder verifier of section 3 is `cross-encoder/stsb-roberta-base`, used off the shelf with no fitting on our pairs.

B Complete results matrix

Table 6 is the full record behind the main-text separability table and the operating-point figures: for every method and domain it gives both threshold-free metrics with their bootstrap intervals, the precision attainable at three recall targets, and the coverage attainable under two false-hit ceilings. The pattern that survives every cut is the adversarial column, where neural and lexical methods alike sit near the base rate; the precision–recall curves behind these numbers are plotted in fig. 9.

Table 6 – Complete reliability matrix for every method and domain: threshold-free separability (PR-AUC, ROC-AUC) with 95% BCA bootstrap intervals, the precision attainable at three recall targets, and the coverage attainable under two false-hit ceilings. This is the full record behind table 3 and fig. 5.

	PR-AUC	ROC-AUC	precision @ recall			coverage @ FHR (%)	
Method	(95% CI)	(95% CI)	0.7	0.8	0.9	≤ 1%	≤ 5%
Questions (base rate 0.36)							
MiniLM-L6	0.76 (0.72,0.78)	0.87 (0.85,0.88)	0.69	0.66	0.61	0.6	0.6
MPNet-base	0.78 (0.75,0.81)	0.89 (0.87,0.90)	0.72	0.70	0.66	0.3	0.3
E5-small	0.74 (0.70,0.77)	0.86 (0.84,0.87)	0.67	0.64	0.60	0.1	0.9
E5-large	0.76 (0.73,0.79)	0.87 (0.85,0.88)	0.69	0.65	0.62	0.6	2.4
BGE-base	0.76 (0.72,0.78)	0.87 (0.85,0.88)	0.70	0.66	0.61	0.6	1.9
BGE-large	0.78 (0.75,0.80)	0.88 (0.86,0.89)	0.71	0.66	0.62	0.5	2.6
TF-IDF (lexical)	0.49 (0.46,0.53)	0.70 (0.68,0.72)	0.51	0.50	0.47	0.0	0.0
Cross-encoder	0.76 (0.72,0.79)	0.87 (0.85,0.88)	0.69	0.67	0.61	0.0	3.6
News (base rate 0.68)							
MiniLM-L6	0.86 (0.84,0.87)	0.76 (0.73,0.78)	0.82	0.79	0.75	1.1	7.4
MPNet-base	0.87 (0.85,0.89)	0.78 (0.76,0.80)	0.84	0.81	0.78	0.5	7.6
E5-small	0.89 (0.88,0.91)	0.81 (0.79,0.83)	0.86	0.82	0.78	8.1	21.7
E5-large	0.90 (0.88,0.91)	0.82 (0.80,0.83)	0.86	0.83	0.79	9.1	24.4
BGE-base	0.88 (0.87,0.90)	0.80 (0.78,0.82)	0.85	0.83	0.79	4.4	18.6
BGE-large	0.89 (0.88,0.91)	0.81 (0.79,0.83)	0.86	0.84	0.80	5.5	19.4
TF-IDF (lexical)	0.84 (0.82,0.85)	0.71 (0.69,0.74)	0.79	0.76	0.72	2.1	5.9
Cross-encoder	0.92 (0.91,0.93)	0.84 (0.83,0.86)	0.88	0.85	0.79	14.8	35.3
Adversarial (base rate 0.44)							
MiniLM-L6	0.59 (0.55,0.62)	0.64 (0.62,0.66)	0.51	0.48	0.45	0.1	0.1
MPNet-base	0.60 (0.57,0.63)	0.66 (0.63,0.68)	0.52	0.49	0.47	0.5	0.9
E5-small	0.54 (0.50,0.57)	0.59 (0.57,0.62)	0.47	0.46	0.45	0.1	0.1
E5-large	0.62 (0.58,0.65)	0.67 (0.64,0.69)	0.53	0.50	0.46	0.2	0.2
BGE-base	0.58 (0.55,0.61)	0.63 (0.61,0.65)	0.50	0.48	0.45	0.1	0.1
BGE-large	0.62 (0.58,0.65)	0.65 (0.63,0.68)	0.51	0.49	0.46	0.3	0.3
TF-IDF (lexical)	0.66 (0.62,0.69)	0.72 (0.70,0.74)	0.59	0.55	0.51	0.2	0.2
Cross-encoder	0.51 (0.48,0.54)	0.60 (0.58,0.63)	0.50	0.48	0.47	0.2	0.2

Precision at a recall target is the cleanliness a cache can hold while still reusing that share of genuinely reusable answers; coverage at a false-hit ceiling is the share of traffic served while keeping errors under the ceiling. The cross-encoder is a verifier (not ANN-indexable), shown for reference.

C Datasheet for the cache-equivalence set

Motivation. The set measures whether an embedding cache's similarity decision matches a human notion of when two prompts may share an answer; the author assembled it for this study.

Composition. Each instance is a pair of short English texts with a binary label: cache-equivalent or not. The pairs are sampled from three public, human-annotated corpora, 2500 pairs each: Quora Question Pairs (duplicates) [16], the Microsoft Research Paraphrase Corpus (news) [4], and PAWS (adversarial word-scrambled paraphrase) [20]. Base rates of the equivalent class are 0.36, 0.68, and 0.44 respectively. Exact-duplicate and empty pairs are removed so that trivial matches do not inflate the positive class.

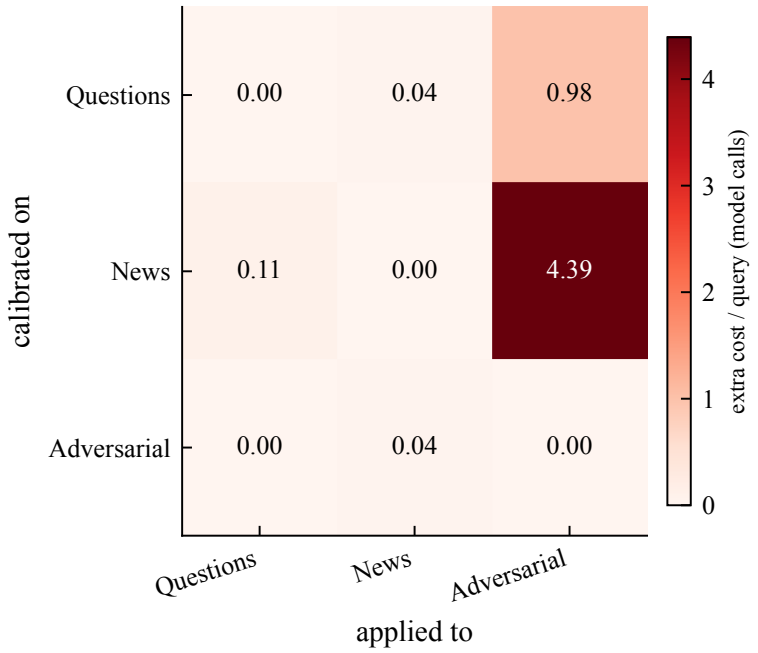


Figure 10 – **Thresholds transfer only in the conservative direction** (section 5). Mean extra cost per query, in model calls, when the cost-aware threshold tuned on one domain (row) is applied to another (column), averaged over encoders at $c_{\text{fh}} = 20$. A threshold tuned on the hard adversarial domain transfers to the easier domains almost for free (bottom row); one tuned on an easy domain is too loose for the adversarial one and backfires, adding up to 4.4 model calls per query (right column). The diagonal is in-domain, zero by construction.

Collection and labeling. Labels are inherited from the source corpora's human annotations; we did not relabel. Sampling is uniform within each corpus under a fixed seed, preserving the natural base rate. The exact sampled pairs are released as CSV files.

Uses and limits. The set is intended for measuring cache reliability and calibrating thresholds. Its labels encode paraphrase and duplication, which proxy answer-equivalence imperfectly (section 7); it should not be read as a gold standard for answer-equivalence in any specific application. The source corpora carry their own licenses, which the released code records; the derived labels and sampling are the author's work.

D Reproducibility

All experiments run on CPU with fixed seeds. The released repository pins dependency versions and provides the exact commands that regenerate every table and figure from the raw scores. The whole pipeline, the reliability sweep over seven methods and three domains, the cross-encoder baseline, and the downstream fault-injection study, runs in well under an hour on a single modern CPU at zero monetary cost; every model runs locally and no paid API is used. A single entry point reproduces it end to end: build the labeled set, score it with each method, compute the bootstrap intervals, run the downstream study, and render the figures.

Table 7 – Each headline claim and the evidence behind it.

Claim	Evidence
Cache-equivalence is distinct from cosine similarity	fig. 1, section 2
On adversarial near-duplicates every encoder is near chance (PR-AUC ≤ 0.62)	table 3, fig. 3
A lexical baseline beats every neural encoder there (paired, $p < 0.003$)	table 3, section 3.1
A joint cross-encoder verifier also collapses there (PR-AUC = 0.51)	table 3
Scale barely helps (+0.08 PR-AUC from small to large E5)	table 3
At a 1% false-hit ceiling, coverage is $\approx$ 9% (news), $<$ 1% (else)	fig. 5, table 6
A false hit costs nearly as much in context as in the answer	fig. 7, section 4
A fixed threshold costs several times more than not caching	table 4
Cost-aware matches the ceiling policy without a hand-set target	table 4, algorithm 1
A threshold transfers across domains only in the conservative direction	section 5

E Pre-specified analysis plan

Before the main run we fixed an analysis plan, released with the code, that named the research questions, the primary metric (PR-AUC under class imbalance, with the false-hit/coverage frontier as the operating-point view), the encoders and domains, the sample size of 2500 pairs per domain, the bootstrap procedure and the multiple-comparison correction, and the rule that all between-encoder comparison would use threshold-free metrics. This is a self-recorded plan in the released materials, not a third-party registration. The sample of 2500 pairs per domain gives several hundred labeled examples per class in every encoder–domain cell, which keeps the bootstrap interval on the primary metric narrow, typically within about ± 0.03 PR-AUC (table 3). Deviations from the plan, if any, are reported with their rationale alongside the code.

F Claims and evidence

Table 7 maps each claim this paper makes to the specific table or figure that supports it, so that a reader can check any one of them directly against the released data.

G Author contributions

As the sole author, Sunny Patel is responsible for all roles under the CREDIT taxonomy: conceptualization, methodology, software, formal analysis, investigation, data curation, writing (original draft and revision), and visualization.