

Semantic Views: Certified Reuse of LLM Labels Across Semantic Queries

Oktie Hassanzadeh

IBM

hassanzadeh@us.ibm.com

Dharmashankar Subramanian

IBM

dharmash@us.ibm.com

Abstract

Semantic query engines evaluate natural-language filters with a large language model, and every evaluation is a costly oracle call. Optimizers that reduce these calls face a dilemma: cascades that trust a cheap proxy miss their precision and recall targets on many queries, methods with finite-sample guarantees save little, and all of them discard the oracle’s labels when the query ends. We present SEMVIEWS, a certified semantic filter that orders tuples by signals that cost no oracle calls (the labels that earlier queries left behind as *semantic views*, a local embedding, and a helper model’s score), refines that order with labels it must pay for anyway, and certifies its answer with fresh stratified oracle samples. The answer meets user precision and recall targets with probability at least $1 - \delta$ for every workload and every catalog, so a stale or misleading view can cost oracle calls but never accuracy. On four workloads replayed against 725,000 recorded oracle labels, SEMVIEWS meets both targets on 99.63% of queries and saves 28%–45% of the oracle calls, whereas the only prior method in our study with the same guarantee needs $1.32\times$ – $1.71\times$ as many calls. When a question returns in different words, views cut its calls by a further 27%–36%, with both targets met on at least 99.8% of such queries; similarity caches and entailment-based reuse meet them on only 17%–47%.

CCS Concepts

• **Information systems** → **Database query processing; Query optimization**; • **Computing methodologies** → *Natural language processing*.

Keywords

semantic operators, LLM-powered query processing, materialized views, result reuse, approximate query processing, statistical guarantees

1 Introduction

Semantic query engines let users filter, join, and rank data with natural-language predicates that a large language model (LLM) evaluates tuple by tuple [7, 31, 39], and cloud data warehouses now expose the same capability as AI functions in SQL [28]. Each evaluation is an *oracle call*, and oracle calls dominate the cost and latency of these queries. A growing body of work therefore optimizes one semantic operator at a time. Model cascades route easy tuples to a cheap helper model and still meet precision and recall guarantees [20, 39, 54]; proxies, clusters, and learned orderings cut the oracle’s share further [5, 18, 57]. These methods face a dilemma. Cascades that trust a proxy’s calibration miss their targets on a large share of queries: on our workloads, the LOTUS cascade misses on 30.8%–65.4% of them. Methods with finite-sample guarantees

keep their targets, but with a weak proxy they save little: BARGAIN [54] with the helper model of the LOTUS cascade saves only 2%–10% of the oracle calls. Across queries, these methods reuse at most query-independent artifacts, such as embedding indexes [22], or thresholds and labels for the *same* predicate. The labels paid for one predicate are never used to answer another.

Example 1.1 (Running example). An analyst explores a table of product reviews and first asks for reviews that “complain about the battery”, then for reviews that “report a hardware defect”, and then for reviews that “say the battery overheats”. The third predicate is narrower than the first, and the second overlaps both. An exact-match cache never hits, because no two predicates are the same. Yet the labels already paid for say a great deal about the third question: a review the oracle judged not to complain about the battery is unlikely to say that the battery overheats.

Semantic queries rarely arrive alone. Analysts refine predicates during exploration, and agents issue such refinements at machine speed [33]. This paper asks whether the labels of earlier predicates can help answer new, different predicates *without weakening the guarantee* that the best per-query optimizers give.

Reuse is tempting, but unsafe. Two kinds of reuse suggest themselves. A similarity cache answers a new predicate with the stored answer of a similar one [59]. Systems that infer containment between predicates, by logic or by an LLM’s entailment judgment [36, 53], reuse an answer when one predicate seems to imply another. Both assume that relations between predicates are crisp. On the oracle’s actual labels, they are not. Across our four workloads, 56% of the implications that an LLM judge asserts are violated by more than 10% of the antecedent’s positives, and containment errors compound along chains of predicates (Section 3). As a result, the answers that similarity caches serve on a hit miss a precision or recall target in 100% of cases in our workloads, and SemWeave’s entailment-based reuse [36] misses on up to 67.8% of queries (Section 7.4).

Our approach. SEMVIEWS is a certified filter that uses everything it has for free to decide where to look, and only fresh oracle samples to decide what is true (Figure 1). Three signals cost no oracle calls: a helper model’s score, a local embedding of each tuple, and, new in SEMVIEWS, the labels that earlier queries left behind. We keep those labels as *semantic views*: label columns bound to a predicate, the columns it reads, and an oracle model, possibly partial, with the provenance of every label. SEMVIEWS fits a score for the new predicate on these signals, refines it with the labels of the tuples it is least sure about, which it would have to pay for anyway, and splits the score order into a reject region, an oracle region, and an accept region. *Fresh* oracle samples, drawn uniformly after the regions are fixed, certify both outer regions with exact finite-sample bounds;

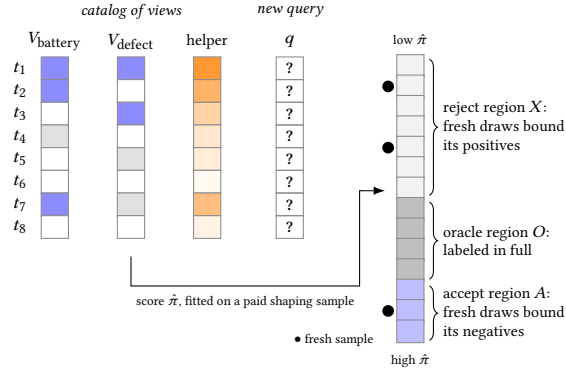


Figure 1: Certified reuse on the running example. Past views (blue: positive, white: negative, gray: unknown) and the helper score are features of a score $\hat{\pi}$ for the new predicate q . The planner splits the score order into three regions. Only fresh oracle samples drawn after the split enter the confidence bounds, so wrong views change the cost, never the guarantee.

the oracle region is labeled in full. The guarantee therefore holds for every workload and every catalog: a stale, partial, or misleading view costs oracle calls, never correctness, and reuse across queries is safe by construction.

Technical challenges. Certified reuse must solve four problems that per-query optimizers do not face. (1) *Untrusted side information.* A view of another predicate can be arbitrarily wrong about the new one, so no bound may depend on view labels being right. (2) *Biased labels.* The labels in a view were chosen by an earlier query’s plan, typically the tuples that were uncertain for a different predicate. They are not a sample of anything for the new predicate, and an estimator that treats them as one is biased. (3) *Data-dependent design.* To profit from views, the regions must depend on the data, yet finite-sample bounds are valid only when the regions are fixed before the draws that certify them. (4) *Planning under uncertainty.* Choosing regions and sample sizes requires predicting whether certification will succeed. A wrong prediction should cost a few extra calls, not validity, and should not force the query to fall back to labeling every tuple.

CERTIFIEDFILTER, the operator at the core of SEMVIEWS, addresses all four by separating *ordering* from *certification*. Views, helper scores, and embeddings enter only a score fitted on paid labels: a small uniform shaping sample, refined with labels of the tuples the score is least sure about (challenges 1 and 2). The planner fixes the regions from that score before any certification draw, and draws without replacement make exact hypergeometric bounds valid for the finite regions (challenge 3). When the planner’s prediction falls short, a deterministic gap-closing step labels the most uncertain tuples, and a single coverage event covers every state in which the operator can stop (challenge 4).

Contributions.

- We present CERTIFIEDFILTER, a semantic filter that meets user precision and recall targets with probability $1 - \delta$

while ordering tuples by free side information, refining that order with labels it must pay for anyway, and certifying two regions with exact finite-sample bounds chosen by a planner that minimizes predicted oracle calls (Section 4).

- We make the oracle’s labels reusable across queries: semantic views with provenance, used only to plan, so that reuse cannot break the guarantee for any catalog, including adversarial ones (Sections 2, 3, and 5).
- We show that uncertified reuse is unsafe: relations between real predicates are statistical and degrade along chains, more than half of the implications an LLM judges fail on the oracle’s labels, and even the same question asked in different words receives different labels (Sections 3 and 7).
- We implement SEMVIEWS as a LOTUS optimizer that rewrites semantic filters without changing the engine, and evaluate it on four predicate pools and a workload of reworded questions against per-query cascades, BARGAIN, exact and similarity caches, and SemWeave’s entailment-based reuse, replayed on recorded oracle labels (725,000 for the four pools alone; Sections 6 and 7).

Summary of results. SEMVIEWS missed a target on 0.37% of 145,000 replayed queries, below δ , and no single predicate exceeded δ . It saved 28%–45% of the oracle calls of exact evaluation, $4\times$ – $18\times$ as much as BARGAIN with the helper score, and every method that is cheaper misses its targets on at least 32.4% of queries. Views add to these savings in proportion to how related a workload’s predicates are: they removed 11%–38% of the calls of a query when an earlier query asked a strongly correlated predicate, and 27%–36% when it asked the same question in different words, where similarity caches and SemWeave miss their targets on 53%–83% of queries; on unrelated predicates they add no measurable cost. Much of the remaining cost is a certification floor that depends on a predicate’s selectivity but not on the size of the relation (Proposition 4.2), so the share of exact evaluation that SEMVIEWS needs falls as tables grow.

Scope and outline. We study semantic filters over text; joins, ranking, and multimodal data are out of scope. Our work is orthogonal to methods that make each call cheaper, such as prefix and KV-cache reuse [32, 48], and to optimizers that allocate error budgets across the operators of one plan [43]: SEMVIEWS shrinks the set of tuples on which a per-query method has to work. Sections 2–6 cover the problem, semantic views, CERTIFIEDFILTER, maintenance, and the system; Sections 7–9 cover the evaluation, discussion, and related work.

2 Background and Problem Statement

Semantic filters. A semantic filter $\sigma_q(R)$ keeps the tuples of a relation R for which a natural-language predicate q holds, as judged by an LLM. Systems such as LOTUS [39] and Palimpsest [31] expose σ_q as an operator over a language expression (“langex”) such as “{review} complains about the battery”, and cloud warehouses expose it as an AI function in SQL [28]. We write $M(t, q) \in \{0, 1\}$ for the label that the oracle model M assigns to tuple t under the system’s prompt for q . Each evaluation of M is an *oracle call*.

Table 1: Notation.

Symbol	Meaning
R, t	relation, tuple
q, p	new predicate, earlier predicate
$M(t, q)$	oracle label of t for q
$\gamma_P, \gamma_R, \delta$	precision and recall targets, failure probability
V_p, C	semantic view of p , catalog of views
$p \sqsubseteq_\varepsilon q$	p is ε -contained in q (Definition 3.1)
K, U	tuples with known / unknown oracle label for q
$\hat{\pi}(t)$	score: predicted probability that $M(t, q) = 1$
X, O, A	reject, oracle, and accept regions of U
n_X, n_A	certification draws from X and A
c_X, c_A	true numbers of positives in X and A
u_X, ℓ_A	upper bound on c_X , lower bound on c_A
E	event that both bounds hold

Reference semantics and targets. Following LOTUS, the exact answer is the reference plan $\sigma_q^*(R) = \{t \in R : M(t, q) = 1\}$, evaluated with one tuple per prompt at temperature zero. An approximate plan returns a set $\hat{\sigma}_q(R)$, judged by its precision $|\hat{\sigma} \cap \sigma^*|/|\hat{\sigma}|$ and its recall $|\hat{\sigma} \cap \sigma^*|/|\sigma^*|$ against the reference. The user supplies a precision target γ_P , a recall target γ_R , and a failure probability δ . A plan *meets its targets* if both measures reach them, and it is *valid* if it meets its targets with probability at least $1 - \delta$ over its own randomness, for every input. Per-query optimizers such as the LOTUS cascade and BARGAIN [54] aim at such targets by scoring every tuple with a cheap proxy and sending only uncertain tuples to the oracle; only methods with finite-sample bounds, such as BARGAIN, are valid in this sense.

Workloads. We model a *workload* as a sequence of semantic filters $q_1, q_2, \dots$ over the same relation and columns, each with its own targets, as in Example 1.1. The predicates differ, so an exact-match cache keyed by predicate and tuple [10, 24] never hits. Yet they are often related: many tuples that pass one predicate also pass another, and many that fail one fail another.

Problem 1 (Certified cross-query reuse). **Input:** a relation R ; the oracle labels paid for by the earlier filters $q_1, \dots, q_{j-1}$ of a workload, organized as a catalog C ; a new filter q_j with targets $(\gamma_P, \gamma_R, \delta)$. **Output:** a set $\hat{\sigma}_{q_j}(R) \subseteq R$. **Constraint:** for every relation, every workload, and every catalog, the output meets both targets with probability at least $1 - \delta$. **Objective:** minimize the expected number of new oracle calls for q_j .

Two features of the problem rule out the obvious solutions. First, validity must hold for every catalog, including one in which every earlier result is useless or misleading for q_j ; reusing a past answer because its predicate looks similar is not valid (Section 7.4). Second, on unrelated workloads the method should cost little more than the best per-query plan, so that enabling reuse is never a gamble. The probability in the constraint is over the method’s own sampling for q_j , given the catalog. Table 1 summarizes our notation.

3 Semantic Views and Statistical Containment

Semantic views. A *semantic view* V_p is a label column over R bound to a predicate p , the columns its langex reads, the oracle

model, and the prompt version. Each entry of V_p is 1, 0, or *unknown*, so views may be partial. Each known entry carries its provenance: *oracle* if it was paid for with a call to $M(t, p)$, or *derived* if it was decided without a call, by accepting or rejecting a whole region under a certificate (Section 4). Every query answered by SEMVIEWS leaves its view behind at no extra cost, because its oracle labels and its region decisions are exactly the view. SemWeave [36] already speaks of treating a prior filter result as a materialized semantic view; we make the notion precise, add provenance and partiality, and, above all, never assume that a view is correct for another predicate.

Containment is statistical. Classical view-based query answering [13, 15] and semantic caching [8] rely on logical containment: if p implies q , every answer to p answers q . Semantic predicates do not behave this way. Whether “says the battery overheats” implies “complains about the battery” is a property of the oracle’s labels on this relation, not of the two sentences, and the oracle is not logically consistent across prompts.

Definition 3.1 (ε -containment). Predicate p is ε -contained in q on R , written $p \sqsubseteq_\varepsilon q$, if

$$\Pr_{t \sim R} [M(t, q) = 0 \mid M(t, p) = 1] \leq \varepsilon.$$

Exclusion is the dual, $\Pr[M(t, q) = 1 \mid M(t, p) = 1] \leq \varepsilon$, and equivalence is containment in both directions. Relations that an LLM judge proposes from the predicate text often fail as statistical ones: 56% of the implications the judge asserts on our four predicate pools have $\varepsilon > 0.1$ on the oracle’s labels (Section 7.2).

Containment does not chain. One might hope to infer relations along a lattice of predicates, as view-matching systems do for logical containment. The following proposition shows that this is unsafe: the error of a chain grows with the ratio of the predicates’ selectivities.

PROPOSITION 3.2 (CHAINED CONTAINMENT). *If $p \sqsubseteq_{\varepsilon_1} q$ and $q \sqsubseteq_{\varepsilon_2} r$ on R , then $p \sqsubseteq_{\varepsilon'} r$ with*

$$\varepsilon' = \varepsilon_1 + \varepsilon_2 \cdot \frac{\text{sel}(q)}{\text{sel}(p)},$$

where $\text{sel}(x) = \Pr_{t \sim R} [M(t, x) = 1]$. The bound is tight whenever $\varepsilon_2 \text{sel}(q) \leq (1 - \varepsilon_1) \text{sel}(p)$.

PROOF SKETCH. Split the event $\{r = 0\}$ on q :

$$\begin{aligned} \Pr[r=0 \mid p=1] &\leq \Pr[q=0 \mid p=1] + \Pr[q=1, r=0, p=1] / \Pr[p=1] \\ &\leq \varepsilon_1 + \Pr[q=1, r=0] / \text{sel}(p) = \varepsilon_1 + \varepsilon_2 \text{sel}(q) / \text{sel}(p). \end{aligned}$$

For tightness, place the ε_1 share of p ’s positives outside q and make them fail r , and place all of q ’s r -failures inside $p \wedge q$, which has room for them under the stated condition. Appendix A.1 gives the full proof. $\square$

When p is narrow and q broad, $\text{sel}(q)/\text{sel}(p)$ is large, and a small error on the second link becomes a large one. For instance, with $\text{sel}(q)/\text{sel}(p) = 10$ and $\varepsilon_1 = \varepsilon_2 = 0.02$, the chain guarantees only $\varepsilon' = 0.22$. Relations between predicates must therefore be established for each pair on the data, never inferred transitively. SEMVIEWS goes one step further and never relies on any relation for

Algorithm 1 CERTIFIEDFILTER ($q, R, C, \gamma_P, \gamma_R, \delta$)

```

1:  $K \leftarrow$  tuples with oracle labels for  $q$  in  $C$ ;  $U \leftarrow R \setminus K$ 
2:  $\mathcal{V} \leftarrow$  the  $k$  views most similar to  $q$  ▷ match
3: label a uniform sample  $S \subset U$ ;  $K \leftarrow K \cup S$ ;  $U \leftarrow U \setminus S$ 
4: fit  $\hat{\pi}$  on  $K$  from  $\mathcal{V}$ , helper scores, embeddings ▷ shape
5: sort  $U$  by  $\hat{\pi}$ ; pick  $X, O, A, n_X, n_A, \alpha_X, \alpha_A$  (Problem 2) ▷ plan
6: for  $r = 1, \dots, r_{\max}$  do ▷ refine
7:   label the  $b$  tuples of  $O$  with  $\hat{\pi}$  closest to  $\frac{1}{2}$ ; refit  $\hat{\pi}$ ; re-plan
8: end for
9: label  $O$ ; draw  $n_X$  from  $X$ ,  $n_A$  from  $A$  without replacement ▷ certify
10:  $u_X \leftarrow U_{\alpha_X}^{\text{HG}}$ ;  $\ell_A \leftarrow L_{\alpha_A}^{\text{HG}}$ 
11: while a condition of Theorem 4.1 fails do ▷ close the gap
12:   label the next unknown tuples at the top of  $X$  or the bottom of  $A$ 
13: end while
14: register  $V_q$ : oracle labels as oracle,  $A/X$  decisions as derived
15: return known positives  $\cup$  ( $A$  minus known negatives)

```

correctness: relations decide where to look, and fresh oracle samples decide what is true.

4 Certified Reuse

SEMVIEWS answers a filter σ_q with the operator CERTIFIEDFILTER (Algorithm 1). The operator obeys one rule: *only fresh oracle labels, drawn uniformly at random from a region fixed before the draws, enter a confidence bound*. Views, helper scores, and embeddings decide how the relation is ordered and split; they never decide what is true. The operator runs in five steps: it matches candidate views (Section 4.1), fits a score on a shaping sample and refines it where it is uncertain (Sections 4.2 and 4.3), plans three regions (Section 4.4), certifies them (Section 4.5), and closes any remaining gap (Section 4.6). Let K be the tuples whose oracle label for q is already known (from an earlier run of q) and $U = R \setminus K$. Labels in K are used exactly and never as samples.

4.1 Matching candidate views

A catalog may hold hundreds of views for a relation. Matching keeps the $k = 10$ views whose predicate text is most similar to that of q , by the cosine similarity of local sentence embeddings, and passes them to the score model as features. Matching does not need to identify the relation between p and q : the score model weights each view by how well its labels predict q 's labels on the shaping sample. This subsumes label co-occurrence and lets a view that is negatively related to q (an exclusion) be as useful as a containing one. Matching errors cost calls, never correctness, because matching only shapes the order. SEMVIEWS records no relation judgments; an LLM judge is used in our evaluation only to drive the SemWeave baseline and to measure how often judged relations fail.

4.2 Shaping: views as features

SEMVIEWS first labels a *shaping sample* of s_0 tuples drawn uniformly from U ($s_0 = 200$ by default, at most 8% of U). These labels are returned exactly and join K , but they never enter a bound. On all labeled tuples, SEMVIEWS fits an L_2 -regularized logistic model $\hat{\pi}(t)$ whose features cost no oracle calls:

- for each candidate view V_p , an indicator that V_p labels t positive and an indicator that V_p does not know t ;

- the helper model's score for q , when a cascade helper is configured, as in LOTUS;
- a 32-dimensional projection of a local sentence embedding of t .

Treating “unknown” as a feature value lets partial views contribute like complete ones. The view indicators are what distinguishes SEMVIEWS from a per-query proxy: past answers to related predicates can be far more predictive of q than a small helper model.

4.3 Refining the score

A score fitted on 200 uniform labels is sample-starved: at 5% selectivity the shaping sample holds about ten positives, too few to learn how much to trust twenty view features. CERTIFIEDFILTER therefore plans once, labels the b tuples of the planned oracle region whose $\hat{\pi}$ is closest to $\frac{1}{2}$, refits, and plans again, for up to $r_{\max} = 8$ rounds. The batch is $b = 5\%$ of the relation; a fixed batch of a few hundred boundary tuples would distort, on a large relation, the probabilities that the planner reads as expected counts. Refinement labels tuples that the current plan would label anyway, uses them exactly, and ends before the regions are fixed and any certification draw is made, so validity is unaffected (Theorem 4.1).

4.4 Planning the regions

Let $u_1, \dots, u_m$ be the tuples of U in ascending order of $\hat{\pi}$. A *plan* (i, j, n_X, n_A) with $i \leq j$ puts the first i tuples in the reject region X , the next $j - i$ in the oracle region O , and the rest in the accept region A ; n_X and n_A are the numbers of certification draws from X and A . The planner predicts what certification would conclude by plugging expected counts into the bounds of Section 4.5. With $\hat{c}_Y = \sum_{t \in Y} \hat{\pi}(t)$ for a region Y , and α the per-region level, it predicts

$$\hat{u}_X = |X| U_{\alpha}^{\text{CP}}(n_X \hat{c}_X / |X|, n_X), \quad \hat{\ell}_A = |A| L_{\alpha}^{\text{CP}}(n_A \hat{c}_A / |A|, n_A),$$

where $U_{\alpha}^{\text{CP}}(x, n)$ and $L_{\alpha}^{\text{CP}}(x, n)$ are the one-sided Clopper–Pearson bounds for x successes in n trials (with x rounded). Let k^+ be the known positives of q . The planner predicts the true positives returned, the positives missed (positives drawn from X are found and returned), and the size of the returned set as

$$\hat{T} = k^+ + \hat{c}_O + \hat{\ell}_A, \quad \hat{\mu} = \max(\hat{u}_X - n_X \hat{c}_X / |X|, 0),$$

and $k^+ + \hat{c}_O + |A|$.

The planner is deliberately pessimistic: it predicts the number of positives among the draws as its expectation shifted by half a standard deviation in the unfavorable direction, which trades a few planned calls for fewer gap-closing rounds. It also chooses how to split δ between the two regions, $\alpha_X = f\delta$ and $\alpha_A = (1 - f)\delta$ with $f \in \{0.5, 0.7, 0.9\}$, because certifying recall on a large reject region usually needs the larger share.

Problem 2 (Region planning). Minimize the predicted oracle calls $|O| + n_X + n_A$ over plans (i, j, n_X, n_A) , subject to $\hat{T} \geq \gamma_R(\hat{T} + \hat{\mu})$ and $\hat{T} \geq \gamma_P(k^+ + \hat{c}_O + |A|)$.

The planner searches boundaries i, j on a grid of $g+1$ evenly spaced positions in $[0, m]$ ($g = 40$) and sample sizes on a doubling grid from 100 to 1,600, using only sizes below 70% of the region. Clopper–Pearson bounds are tabulated once per sample size, so each candidate plan is evaluated in constant time and the search costs $O(g^2 \cdot$

$5^2 \cdot 3$ evaluations (the last factor for the δ split), plus $O(m \log m)$ for sorting. If no plan is predicted to cost fewer calls than $|U|$, everything is O and the operator degenerates to exact evaluation. The plan depends on the catalog, the helper, the embeddings, and the shaping labels, but on no label drawn later.

4.5 Certification

SEMVIEWS labels O in full. It then draws n_X tuples from X and n_A from A uniformly *without replacement*, so the number of positives among the draws from a region is hypergeometric with the region’s (unknown, fixed) number of positives as its parameter. One-sided exact hypergeometric bounds at levels α_X and α_A with $\alpha_X + \alpha_A = \delta$ (or δ for the only non-empty region) are then valid for the finite region without any approximation: U^{HG} is the largest count c for which observing at most the drawn positives has probability above α_X , and L^{HG} is defined symmetrically. They are markedly tighter than Clopper–Pearson bounds [6] on the same draws when the sample is a sizable share of the region. While planning, SEMVIEWS approximates them by Clopper–Pearson bounds with a finite-population correction. Let u_X be the resulting upper bound on the number of positives in X , ℓ_A the lower bound on the number of positives in A , and E the event that both bounds hold; by a union bound, $\Pr[E] \geq 1 - \delta$.

SEMVIEWS returns every known positive and every tuple of A except known negatives. To state when this answer is certified, consider any state of the algorithm after certification and define:

- P : known positives outside A (in K , in O , and those found in X);
- d_X and v_X : known positives and still-unknown tuples in X ;
- d_A and e_A : known positives and known negatives in A ;
- $T = P + \max(\ell_A, d_A)$, a lower bound on the true positives returned;
- $\mu = \min(\max(u_X - d_X, 0), v_X)$, an upper bound on the positives missed.

THEOREM 4.1 (VALIDITY). *On the event E , if $T \geq \gamma_R(T + \mu)$ and $T \geq \gamma_P(P + |A| - e_A)$, the returned set has recall at least γ_R and precision at least γ_P . Hence CERTIFIEDFILTER meets both targets with probability at least $1 - \delta$, for every relation, workload, and catalog.*

PROOF SKETCH. Let c_X and c_A be the true numbers of positives in X and A . The returned set contains $P + c_A$ true positives (no positive of A is dropped), has size $P + |A| - e_A$, and misses $c_X - d_X \leq v_X$ positives. Recall $(P + c_A)/(P + c_A + c_X - d_X)$ increases in c_A and decreases in c_X ; precision $(P + c_A)/(P + |A| - e_A)$ increases in c_A . On E , $c_A \geq \ell_A$ and $c_X \leq u_X$, and trivially $c_A \geq d_A$, so T and μ yield lower bounds on both ratios. The argument holds simultaneously for every state in which the algorithm can stop, so data-dependent stopping is covered. Views enter only through $\hat{\pi}$, which fixes X and A before the draws. Appendix A.2 gives the full proof and its extension to any number of strata. $\square$

4.6 Closing the gap

The planner works with predictions, so a condition of Theorem 4.1 may fail after certification. SEMVIEWS then labels the highest-scored

unknown tuples of X in batches of 50 (each positive found is returned and lowers the bound on missed positives) or the lowest-scored unknown tuples of A (each negative found is dropped), and checks again. Labeling a region completely makes its contribution exact, so the loop always terminates with both conditions satisfied. Every intermediate state is covered by the same event E , so gap closing never affects validity; it turns a near miss of the planner into a few extra calls instead of a failed query or a fall back to exact evaluation.

4.7 Discussion

Why not trust the views? A view V_p with $p \sqsubseteq_\varepsilon q$ suggests accepting p ’s positives for q , and $q \sqsubseteq_\varepsilon p$ suggests rejecting p ’s negatives. On real predicates ε is rarely small (Table 3), and errors compound along chains (Proposition 3.2), so a system that reuses on the strength of an inferred relation misses its targets (Section 7.4). CERTIFIEDFILTER uses the same information to place the regions and pays a few hundred fresh labels to certify them.

A certification floor. Any sampling-based guarantee pays a per-query cost that depends on the predicate’s selectivity but not on the size of the relation, and on small relations this floor, not the quality of the score, limits the savings.

PROPOSITION 4.2 (CERTIFICATION FLOOR). *Let q have $c = sN$ positives among N tuples, and let a plan reject $|X| = xN$ tuples. With Clopper–Pearson bounds, even if no certification draw from X is positive, recall γ_R is certified only if*

$$n_X \geq \frac{\ln(1/\alpha)}{-\ln(1-\rho)} \approx \kappa \frac{x}{s}, \quad \rho = \frac{(1-\gamma_R)s}{\gamma_R x}, \quad \kappa = \frac{\gamma_R \ln(1/\alpha)}{1-\gamma_R},$$

unless all but at most $(1-\gamma_R)c/\gamma_R$ tuples of X are labeled. The plan then costs at least $|O| + n_X \geq (1-x-a)N + \kappa x/s$ calls, where $a = |A|/N$.

PROOF SKETCH. Hypergeometric bounds lower the floor noticeably only when n_X is a sizable share of $|X|$. With no positive among n_X draws, the Clopper–Pearson bound is $u_X = |X|(1 - \alpha^{1/n_X})$ and $d_X = 0$. Theorem 4.1 needs $\mu \leq (1 - \gamma_R)T/\gamma_R$, and $T \leq c$, so $|X|(1 - \alpha^{1/n_X}) \leq (1 - \gamma_R)c/\gamma_R$; solving for n_X gives the bound. Positive draws only raise u_X . $\square$

For $\gamma_R = 0.9$ and $\alpha = \delta/2 = 0.025$, $\kappa \approx 33$: a predicate with 1% selectivity needs about 3,300 draws whenever most tuples are rejected, which is two thirds of a 5,000-tuple relation but a rounding error on a relation of millions. The bound holds for every score, including a perfect one. Any method that certifies recall from a uniform sample of the rejected tuples faces a floor of the same order, so per-query cascades with guarantees pay it as well.

How CERTIFIEDFILTER differs from a per-query cascade. Per-query cascades [20, 39, 54] threshold one scalar proxy and estimate the quality of the threshold from samples of the whole relation. CERTIFIEDFILTER differs in three ways. Its score combines free past labels of other predicates with the usual proxies. Every label it already holds, from an earlier run of q , the shaping sample, the oracle region, or gap closing, enters the precision and recall accounting exactly rather than through an estimator. And it certifies two regions with

exact bounds chosen by a planner that trades the size of the oracle region against the number of certification draws. With views disabled, CERTIFIEDFILTER is a finite-sample stratified cascade; we report that configuration separately so that the effect of views and the effect of the certifier can be told apart (Section 7.3).

5 Maintaining Semantic Views

Views are a by-product of answering queries, so maintaining them must be cheap and must never put the guarantee at risk. SEM-VIEWS meets both requirements with one principle: a view is never assumed correct, so every change to the data or to the oracle is handled by re-certifying, not by recomputing.

Registering a view. When CERTIFIEDFILTER finishes, it registers the view of q : every oracle label with provenance *oracle*, and the unlabeled tuples of A and X as derived positives and negatives. If a view of q already exists, the two are merged tuple by tuple: an oracle label always wins, and a derived label only fills an entry that has no oracle label. Only oracle labels of q itself are used exactly when q is asked again; derived labels, like views of other predicates, only shape the score. A repeated predicate is therefore answered with the labels it already paid for, which subsumes an exact-match cache without trusting any derived decision.

Appends. Rows are identified by a content hash of the columns that the predicate reads (Section 6). A new row enters every view as *unknown*, which is already a feature value of the score model. The view remains usable, and the score model learns from the shaping sample how to order new rows. No view is invalidated and no label is recomputed. We measure this case in Section 7.6.

Oracle upgrades. A view’s identity includes the oracle model and the prompt version. After an upgrade, queries under the new oracle find no oracle labels of their own, so nothing from the old oracle is used exactly. Views recorded under the old oracle remain usable as features, because Theorem 4.1 never assumes that features are correct. Section 7.6 shows that such views still save calls.

Storage and lifecycle. In memory, a view costs two bytes per tuple: one for the label and one for the provenance. On disk, the catalog stores only known entries. All our experiments keep every view. Two lifecycle questions remain open: which partial views to complete during idle time, and which views to drop under a storage budget. Both are selection problems over the workload history, in the spirit of classical view selection [16]; we leave a budgeted advisor to future work.

6 System

SEMVIEWS is a Python package with an engine-independent core and a thin adapter for LOTUS [1, 39] (Figure 2). The core contains the catalog, view matching, the certification bounds, and CERTIFIEDFILTER. It does not import LOTUS, and a test enforces this separation, so adapters for other engines need only supply an oracle, optional helper scores, and row identities.

LOTUS integration. The adapter is a LOTUS LazyFrame optimizer. It rewrites each SemFilterNode into a ViewFilterNode that keeps every field of the original node and is bound to a catalog; every other node, prompt, and model is left unchanged, and no

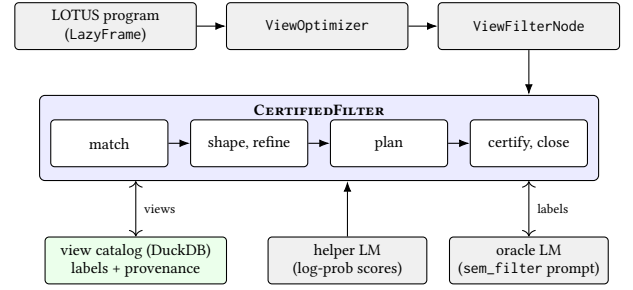


Figure 2: Architecture. The LOTUS adapter (gray) rewrites semantic filters into nodes that run CERTIFIEDFILTER (blue) against a persistent view catalog. Oracle calls use LOTUS’s own prompt and model, so answers keep LOTUS’s reference semantics.

LOTUS source is modified. Users opt in by adding one optimizer to a program:

```

catalog = Catalog("reviews", 0, path="reviews.views.duckdb")
args = CascadeArgs(recall_target=0.9, precision_target=0.9,
                   failure_probability=0.05)
lf = LazyFrame().sem_filter("{text} says the battery overheats",
                           cascade_args=args)
out = lf.optimize([ViewOptimizer(catalog)]).execute(df)

```

The targets are read from LOTUS’s own CascadeArgs. The oracle is LOTUS’s own sem_filter prompt on the configured model, and helper scores are LOTUS’s own log-probability scores from its helper model, so SEMVIEWS answers with respect to exactly the reference semantics that LOTUS defines. Filters with options that SEMVIEWS does not support (explanations, raw outputs, few-shot examples, or a non-default reasoning strategy) run through the native LOTUS path.

Identities and catalog. A row’s identity is a content hash of the columns that the langex reads, so views survive across DataFrames and sessions, and rows of a new DataFrame that the catalog has not seen are appended to every view as unknown. A view’s identity is a hash of the langex, the prompt version, and the oracle model. The catalog is a DuckDB file beside the data with four tables: the table’s metadata, the row identities, one row per view (predicate text, model, and the targets under which it was produced), and the known labels with their provenance, stored sparsely.

Cost of the operator itself. Besides oracle calls, CERTIFIEDFILTER embeds the tuples locally and, per refinement round, fits a small logistic model and runs the planner’s grid search: a median of 849 ms per query in our single-threaded implementation.

7 Evaluation

We answer six questions, one per subsection. (1) Do real workloads contain the relations that reuse needs, and can relations judged by an LLM be trusted? (2) How many oracle calls does SEMVIEWS save, and where do the savings come from? (3) Do the guarantees hold, and do uncertified alternatives keep them? (4) Is reuse safe and profitable when a question returns in other words? (5) How robust is SEMVIEWS to unrelated workloads, adversarial or partial

Table 2: Datasets and predicate pools. Selectivity is the median share of tuples that the oracle labels positive, with the range over predicates in parentheses.

Dataset	Text	Predicate pool	Preds	Selectivity
PubMed	Biomedical abstracts	Human-written, external	25	13% (0.7–51%)
GoEmotions	Social-media comments	From emotion taxonomy	40	23% (3.0–57%)
DBpedia	Encyclopedia entries	From class taxonomy	40	5% (0.9–59%)
Reviews	Product reviews	Analyst-style, ours	40	7% (0.2–78%)

views, appends, and oracle changes? (6) How sensitive is it to its parameters and components?

7.1 Experimental Setup

Datasets and workloads. Table 2 summarizes the four text relations, each a sample of 5,000 tuples drawn with a fixed seed. Measuring validity requires the oracle’s label for every tuple and predicate, so we limit the relations’ size to limit the cost and carbon footprint of the experiments; larger-scale experiments (Appendix B) confirm that the findings hold on larger relations. *PubMed* pairs biomedical abstracts with the 25 human-written predicates that ScaleDoc [56] released for this corpus, used verbatim; it is our external workload. *GoEmotions* and *DBpedia* (social-media comments and encyclopedia entries) each have 40 predicates generated mechanically from the datasets’ published label taxonomies (emotions, Ekman groups, and sentiments; three levels of DBpedia classes), so their logical containments are known by construction. *Reviews* has 40 analyst-style predicates over electronics product reviews (battery, defects, shipping, service, sentiment, ...), written before any label was seen. A workload issues all predicates of a pool in a random order. For each pool we also build a control workload of eight nearly uncorrelated predicates. Finally, a *reworded* workload models users and agents that ask the same question again in other words: for 15 predicates drawn at random from each of the *Reviews* and *GoEmotions* pools, gpt-oss-120b wrote one rewording that keeps the meaning, which the oracle labeled like every other predicate; the workload issues each pool together with its rewordings in random order.

Oracle, helper, and recorded labels. The oracle is Llama 3.3 70B Instruct at temperature zero, prompted with LOTUS’s own `sem_filter` prompt, one tuple per prompt. The helper is Llama 3.2 3B Instruct (4-bit) served locally; its score is LOTUS’s own log-probability score. We recorded every (tuple, predicate) oracle label and helper score once, 725,000 oracle calls in all, and replay every method against this record. Oracle-call counts are therefore exact, and validity can be checked over hundreds of workload orders. Re-labeling 2,000 random pairs twice reproduced the recorded label in 99.85% of cases. Because the oracle is not perfectly deterministic, the reference answer, and hence every precision, recall, and validity number, is defined with respect to the recorded labels, as in LOTUS and Larch [57]. A second oracle, gpt-oss-120b, labels the first 1,000 *GoEmotions* tuples for the oracle-change experiment. Row and predicate embeddings come from a local sentence encoder.

Methods. *Exact evaluation* labels every tuple. The per-query methods with targets are:

- *LOTUS cascade*: LOTUS 1.2.4’s own importance sampling, calibration, and threshold code on the recorded helper scores, with its default `CascadeArgs` (a 10% importance sample drawn from the first 200 tuples, 50 calibration quantiles); and a *full-range* variant whose importance sample spans the whole relation;
- *BARGAIN* [54]: its released code, with joint precision and recall targets, on the helper score;
- *embedding proxy*: logistic regression on row embeddings trained on a 5% labeled sample and thresholded with the LOTUS cascade, following [5];
- *SEMViews, no views*: CERTIFIEDFILTER with only helper and embedding features.

The cross-query methods are:

- *exact cache*: no predicate repeats, so it equals its LOTUS-cascade fallback and is omitted from the tables;
- *similarity cache*: reuses the labels of the most similar past predicate when the cosine similarity of the predicate texts exceeds $\tau \in \{0.7, 0.8, 0.9\}$, and otherwise runs the LOTUS cascade; a *similarity proxy* instead feeds those labels to the LOTUS cascade as proxy scores;
- *SemWeave* [36], reimplemented from its Algorithm 1: for each new filter it retrieves the five most similar past filters, obtains their bidirectional entailment relation from an LLM judge (one prompt per pair, with SemWeave’s consistency check), and decides a tuple without the oracle whenever a stored result and the relation imply its label;
- *SEMViews: CERTIFIEDFILTER with views*.

SemWeave has no precision or recall targets; we report how often its answers would meet ours. Unless stated otherwise, $\gamma_P = \gamma_R = 0.9$ and $\delta = 0.05$.

Metrics and protocol. We report oracle calls and the share of queries that miss either target. Costs are medians over 20 workload orders. Validity is measured over 1,000 orders for SEMViews (145,000 queries) and 200 orders for the baselines. We count calls rather than dollars because the two agree: every recorded call carries its token counts, and for every method and dataset, the share of exact evaluation’s dollar cost differs from the share of its calls by at most 0.7 percentage points.

7.2 Relations Among Real Predicates

Containment is real but rarely tight, and judged implications usually fail. Table 3 measures the containment error $\varepsilon = \Pr[q=0 \mid p=1]$ on the oracle’s labels for every ordered pair of predicates over the same relation. Across the four pools, only 92 of 5,280 ordered pairs are 0.1-contained, while near-exclusion is common. The LLM judge called 146 pairs implications, and 56% of those have $\varepsilon > 0.1$. Implications derived from a taxonomy fare no better, because the oracle is not logically consistent across prompts. This is the failure that uncertified reuse inherits.

7.3 Oracle Calls

SEMViews is the cheapest valid method on every dataset. Figure 3 plots cumulative oracle calls as each workload unfolds. SEMViews saves 28%–45% of the calls of exact evaluation. BARGAIN with the

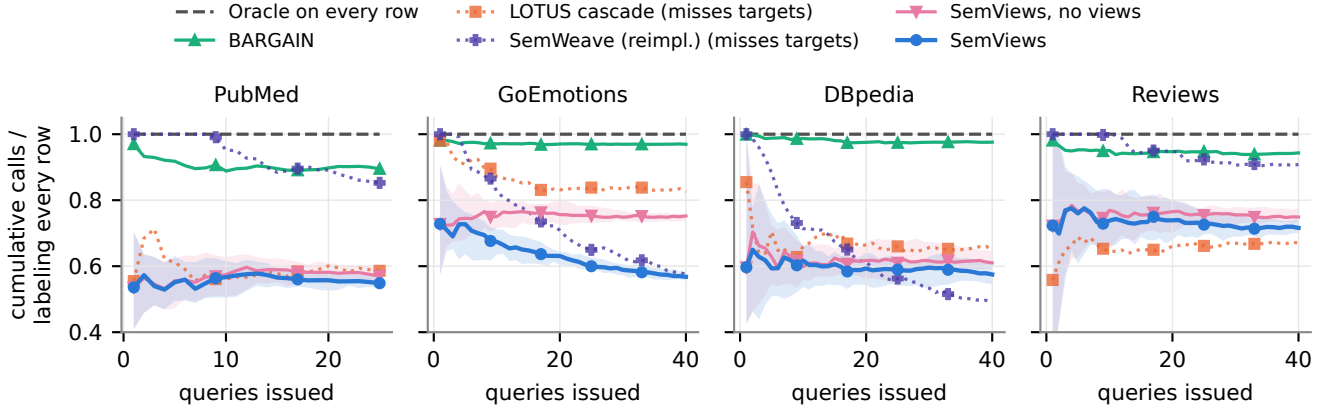


Figure 3: Cumulative oracle calls as a share of exact evaluation as each workload unfolds (median over workload orders; bands: 10th to 90th percentile). Dotted methods miss their targets on many queries (Table 5); SEMVIEWS is the cheapest method that meets them.

Table 3: Containment and exclusion among predicates on the oracle’s labels: ordered pairs that are ε -contained ($\sqsubseteq_\varepsilon$) or ε -exclusive (Excl_ε), and implications asserted by an LLM judge, with the share that fail statistically ($\varepsilon > 0.1$).

Dataset	Preds	$\sqsubseteq_{0.05}$	$\sqsubseteq_{0.1}$	$\text{Excl}_{0.05}$	Judged $\Rightarrow$ (fail)
PubMed	25	1	5	123	12 (58%)
GoEmotions	40	12	29	268	58 (64%)
DBpedia	40	29	37	1107	62 (45%)
Reviews	40	11	21	672	14 (71%)

helper score, the only prior per-query method that meets its targets on every dataset, saves 2%–10%: SEMVIEWS saves 4×–18× as much, and BARGAIN needs 1.32×–1.71× the calls of SEMVIEWS. Against the cheapest per-query method that meets its targets on at least 95% of queries (in every case SEMVIEWS without views), the factor is 1.04×–1.32×. The LOTUS cascade and SemWeave appear cheaper on some datasets, but they do not meet their targets (Section 7.4).

Later queries. The cumulative totals include the first queries of each workload, which find an empty catalog. Over the last ten queries, SEMVIEWS needs as little as 52% of the calls of exact evaluation, and BARGAIN needs 1.32×–1.89× the calls of SEMVIEWS. Views remove up to 31% of the calls of the operator without them on GoEmotions, the pool richest in related predicates; on the other pools, their late gain stays close to their workload average (Table 4).

Views help when the workload holds related predicates. Against the same operator with views disabled, views remove 4%–25% of the calls over a whole workload, but this average mixes two kinds of queries (Table 4). When an earlier query asked a strongly correlated predicate, views remove 11%–38% of the calls; otherwise they remove at most 14%, and at most 5% outside GoEmotions, where overlapping emotions also help weakly correlated predicates. The workload average therefore follows how often a pool contains related predicates: 25 of 40 GoEmotions predicates have

Table 4: Calls saved by views, relative to SEMVIEWS without views on the same workload order. *Related preds*: predicates whose oracle labels correlate with another predicate of the pool ($|\phi| > 0.6$). *Related and unrelated*: queries for which an earlier query of the order did or did not ask such a predicate. *All*: the whole workload; *last 10*: its last ten queries.

Dataset	Related preds	Calls saved by views			
		related	unrelated	all	last 10
PubMed	4/25	11%	3%	4%	5%
GoEmotions	25/40	38%	14%	25%	31%
DBpedia	9/40	19%	5%	6%	2%
Reviews	9/40	30%	1%	4%	4%

a strongly correlated partner, and at most 9 in each other pool. On DBpedia, the embedding alone already separates most topical classes, so even related views add less. Refinement matters here: the first-draft operator, which fits its score on the 200-tuple shaping sample alone, gained only 2%–31% from related views, because so few labels cannot teach the score how much to trust twenty view features.

Certification, features, and refinement contribute separately. To isolate the certifier, we give BARGAIN the same shaping sample and the same score that SEMVIEWS fits on it before refinement. BARGAIN’s threshold search then needs 0.99×–1.18× the calls of region certification on that score (the first-draft operator of Table 9), and refining the score removes another 8%–21% of the calls.

Most calls label tuples the score cannot settle. Figure 4 splits the calls of CERTIFIEDFILTER by step. Refinement rounds and the oracle region, which both label the tuples the score is unsure about, account for 41%–54% and 6%–11% of the calls, certification draws for 18%–32%, gap closing for 6%–20%, and the shaping sample for 6%–7%. Views reduce cost by sharpening the score, so that fewer tuples need an exact label before the planner can place the rest in the

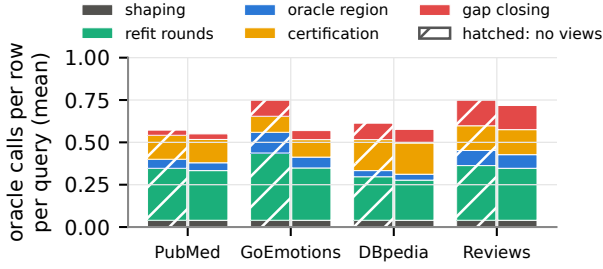


Figure 4: Where the oracle calls go: mean calls per tuple and query, split by the step of CERTIFIEDFILTER that spends them, without views (hatched) and with views. Refinement and the oracle region label the tuples the score cannot settle; views shrink both.

Table 5: Share of queries that miss a target ($\gamma_P = \gamma_R = 0.9$, $\delta = 0.05$). Bold: above δ . Only SEMVIEWS, its view-free variant, and BARGAIN are valid on every dataset.

Method	PubMed	GoEmotions	DBpedia	Reviews
SemViews	0.0%	0.4%	0.2%	0.7%
SemViews, no views	0.0%	0.9%	0.3%	0.9%
BARGAIN	0.5%	0.6%	0.2%	0.3%
LOTUS cascade	53.5%	30.8%	65.4%	32.4%
LOTUS cascade (full-range)	53.1%	10.4%	57.5%	24.1%
Embedding proxy	49.0%	65.4%	47.3%	54.2%
Similarity proxy	52.9%	35.0%	65.3%	35.8%
SemWeave (reimpl.)	16.3%	47.6%	67.8%	6.2%
Similarity cache (0.7)	73.0%	60.5%	80.3%	36.4%
Similarity cache (0.8)	66.0%	42.9%	73.3%	34.9%
Similarity cache (0.9)	59.1%	30.8%	67.7%	32.4%

certified regions. The cost of certification itself is nearly constant, so its share falls as relations grow (Section 7.7).

Rare predicates sit at the certification floor. Cost depends strongly on selectivity. Predicates with at most 2% selectivity cost 80% of exact evaluation, those above 10% cost 54%. This is the floor of Proposition 4.2: on 5,000 tuples, certifying recall for a rare predicate needs a sample comparable to the relation itself, whatever the score. Of the 145 predicates, 44 have at most 5% selectivity, so the floor weighs on every average we report, most on DBpedia and Reviews, whose median selectivities are lowest (Table 2).

7.4 Validity

SEMVIEWS meets its guarantee on every dataset and every predicate. Table 5 reports the share of queries that miss either target. SEMVIEWS misses on at most 0.7% of queries on any dataset, below $\delta = 0.05$, as Theorem 4.1 requires. Because the guarantee is per query, we also check each predicate separately: across 145 predicates, the worst per-predicate miss rate over its 1,000 runs is 3.2% (95% upper confidence bound 4.5%), and 0 predicates exceed δ . BARGAIN is valid as well (at most 0.6%).

The LOTUS cascade misses often, even with stricter internal targets. The LOTUS cascade misses on 30.8%–65.4% of queries, and

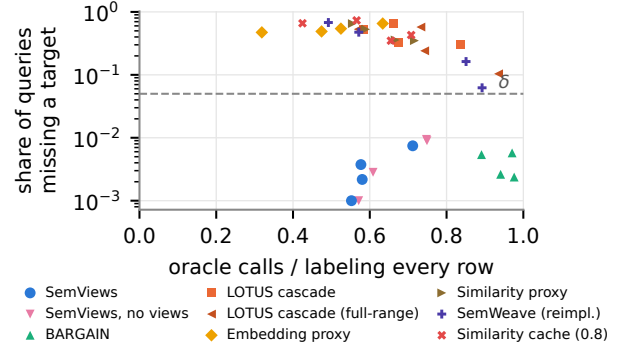


Figure 5: Cost against validity. Each point is one method on one dataset. Only methods below the dashed line meet the guarantee; among them, SEMVIEWS is the cheapest.

Table 6: Reworded workload: oracle calls (share of exact evaluation) and share of queries that miss a target, for queries whose twin, the same question in other words, was asked earlier (calls: 20 workload orders; misses: 200). Bold: above δ .

Method	Reviews		GoEmotions	
	Calls	Miss	Calls	Miss
SemViews	0.53	0.0%	0.47	0.2%
SemViews, no views	0.73	0.7%	0.74	0.5%
BARGAIN	0.91	0.3%	0.97	0.5%
LOTUS cascade	0.57	43.3%	0.82	26.9%
SemWeave (reimpl.)	0.07	68.3%	0.06	82.5%
Similarity cache (0.8)	0.38	52.9%	0.24	63.0%
Similarity cache (0.9)	0.55	46.2%	0.79	27.6%

102 of 145 predicates exceed δ . Its thresholds come from normal approximations on an importance sample drawn from the first 200 tuples; the full-range variant still misses on up to 57.5%. Raising its targets by 0.08 and setting $\delta = 0.001$ does not fix this: it still misses on 1.3%–48.0% of queries, is valid on only 1 of four datasets, and costs nearly as much as exact evaluation.

Uncertified reuse is not safe. Similarity caches miss on 31%–80% of queries. Part of these misses comes from their fallback, the LOTUS cascade, but when the cache actually hits, the answer misses a target 100% of the time. SemWeave misses on 6.2%–67.8% of queries at 49%–91% of the cost of exact evaluation. It reuses stored results on 56% of queries and misses on 65% of those, with a 5th-percentile recall of 0.22: entailment judged from the predicate text is not entailment on the oracle’s labels. The embedding proxy, which trusts a learned classifier without certifying it, misses on up to 65.4%.

Cost and validity together. Figure 5 combines both measures. On every dataset, each method that is cheaper than SEMVIEWS misses its targets on at least 32.4% of queries, and each method that meets the guarantee costs more than SEMVIEWS.

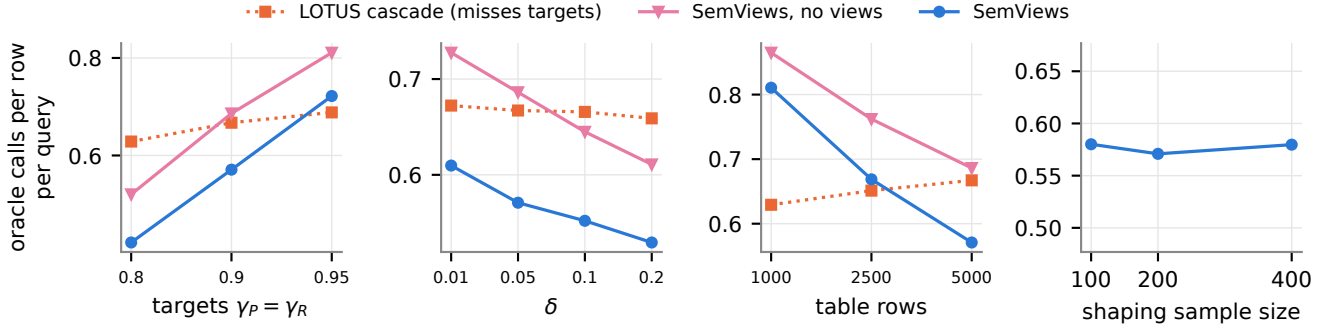


Figure 6: Oracle calls per tuple and query (median over all four datasets and workload orders) as the targets, δ , the table size, and the shaping-sample size vary one at a time. Larger tables help because the certification floor does not grow with the table.

Table 7: Oracle calls on control workloads of nearly uncorrelated predicates (median over workload orders). Views cost almost nothing when they cannot help.

Dataset	Preds	No views	SemViews	Overhead
PubMed	8	22,818	22,823	+0.0%
GoEmotions	8	34,055	32,895	-3.4%
DBpedia	8	25,848	25,400	-1.7%
Reviews	8	26,500	25,431	-4.0%

Table 8: Views recorded under one oracle and reused under another (GoEmotions, 1,000 tuples): total oracle calls and share of queries that meet both targets.

Method	Same-oracle views		Views from another oracle	
	Calls	Met	Calls	Met
SemViews	32,909	99%	29,526	100%
SemViews, no views	36,026	99%	36,026	99%
LOTUS cascade	30,276	63%	30,276	63%
Similarity cache (0.9)	30,276	63%	0	0%

7.5 Repeated Questions in Different Words

The oracle does not answer a rewording the same way. A reworded predicate agrees with its original on 76%–99.8% of the tuples, but because most tuples are negatives, the overlap of the two positive sets (Jaccard) is only 0.57–0.95. Reusing the twin’s answer therefore breaks a 90% precision or recall target more often than not: SemWeave, whose judge correctly calls almost every twin equivalent, and the similarity caches miss their targets on 53%–83% of these queries (Table 6).

Certified reuse turns the twin into savings and keeps the guarantee. SEMVIEWS uses the twin’s view only to order the tuples. It needs 47%–53% of the calls of exact evaluation on repeated questions, 27%–36% fewer than without views, and misses its targets on at most 0.2% of them over 200 workload orders.

7.6 Robustness

Unrelated workloads cost nothing measurable. On the control workloads of eight nearly uncorrelated predicates, views add no measurable cost (Table 7; worst case +0.0%) and reduce calls slightly on three of the four pools. The shaping sample is needed anyway, and an uninformative view only adds features that the score model learns to ignore.

Adversarial and inverted views do not break the guarantee. We seed the catalog with a misleading view for every predicate: the same predicate text, so matching ranks it first, but labels drawn at random with the predicate’s selectivity. SEMVIEWS still meets its targets on at least 99.0% of queries and costs no more than the

view-free operator (cost ratio at most 0.98, because the workload’s genuine views still accumulate). A similarity cache that trusts the matching text meets its targets on 0% of queries. A view whose labels are the *inverse* of the predicate’s (with 20% noise) is not misleading at all for SEMVIEWS: the score model learns a negative weight, and calls drop by 16%–40%.

Appends. We record views for every predicate on the first 80% of the tuples, append the remaining 20%, and run the workload again. The new tuples are unknown in every view. SEMVIEWS meets its targets on 100.0% of queries and needs 32%–49% of the calls of exact evaluation, far less than in a fresh workload because every predicate already has a view. A similarity cache that reuses the stale labels meets its targets on 0% of queries.

Views survive an oracle change. We answer GoEmotions queries under a second oracle, gpt-oss-120b, with views recorded under Llama 3.3 (Table 8). SEMVIEWS meets its targets on 100% of queries and saves 18% of the calls of the view-free operator, more than with views from the same oracle, because all views exist from the first query. A similarity cache that reuses the old oracle’s labels for the same predicate meets its targets on 0% of queries.

Partial views. Figure 7 keeps a random share of each view’s labels. On GoEmotions, where views help most, savings grow steadily with coverage; half of each view still saves 13% of the calls of the view-free operator. On the other datasets, partial views keep the cost within 3.6% of the view-free operator, sometimes slightly above it, because sparse views add features that carry little signal.

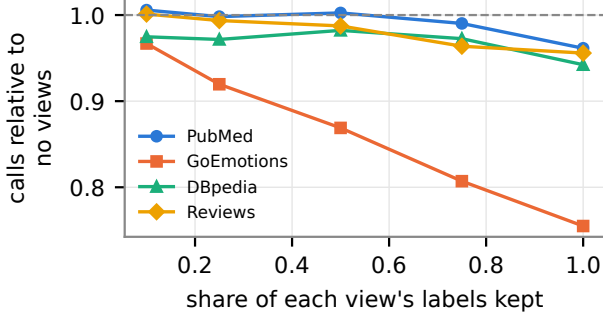


Figure 7: Oracle calls relative to the view-free operator as each view keeps only a random share of its labels. Savings grow with coverage where views are informative; sparse views cost little.

Table 9: Ablations: oracle calls of each variant relative to SEMVIEWS on the same workload order, median and [10th, 90th percentile] over orders (lower is better).

Variant	PubMed	GoEmo.	DBpedia	Reviews
first draft (no refit, CP)	1.23 [1.18,1.27]	1.16 [1.14,1.19]	1.27 [1.19,1.33]	1.08 [1.04,1.12]
no views	1.04 [1.00,1.08]	1.32 [1.29,1.35]	1.04 [0.99,1.16]	1.03 [1.02,1.08]
no views, no embeddings	1.50 [1.48,1.55]	1.61 [1.57,1.66]	1.67 [1.57,1.76]	1.22 [1.19,1.27]
per-signature strata	1.61 [1.57,1.66]	1.42 [1.35,1.44]	1.64 [1.52,1.71]	1.23 [1.21,1.29]
no row embeddings	1.38 [1.35,1.40]	1.18 [1.15,1.22]	1.51 [1.43,1.57]	1.15 [1.10,1.19]
no helper scores	1.11 [1.06,1.14]	1.01 [0.99,1.04]	0.99 [0.90,1.04]	0.97 [0.93,1.00]
all views as candidates	1.00 [0.98,1.05]	0.99 [0.96,1.01]	1.01 [0.95,1.06]	0.98 [0.95,1.03]
shaping sample 100	1.02 [0.98,1.06]	1.02 [0.99,1.04]	1.02 [0.94,1.11]	1.00 [0.96,1.06]
shaping sample 400	1.01 [0.97,1.05]	1.02 [1.00,1.05]	0.99 [0.93,1.09]	0.99 [0.95,1.02]

7.7 Sensitivity, Ablations, and Overheads

Targets, δ , and table size. Figure 6 varies one parameter at a time. Costs rise with stricter targets and fall slowly with larger δ , since the bounds depend on $\log(1/\delta)$. Larger tables help: the certification floor does not grow with the table (Proposition 4.2), so its share of the calls falls as tables grow, and the operator’s cost falls from 81% of exact evaluation at 1,000 tuples to 57% at 5,000 (median over datasets and orders).

Ablations. Table 9 removes or replaces one component at a time, pairing each variant with SEMVIEWS on the same workload order. Bounding each view-signature stratum separately (the per-signature strata variant) costs substantially more: real strata are not pure enough, and per-stratum bounds dilute δ . Row embeddings are the most valuable free signal; without them the operator is markedly more expensive on every dataset. The helper’s score helps on PubMed and barely matters elsewhere. Considering all views instead of the top ten barely changes the cost. With refinement, the size of the shaping sample hardly matters: refinement rounds supply the labels that a larger sample would. The first-draft operator, without refinement and with Clopper–Pearson bounds on draws with replacement, needs $1.09\times$ – $1.27\times$ the calls of SEMVIEWS.

Latency. All methods replay the same recorded labels, so we report latency through its two drivers. First, the operator’s own

computation (Section 6) takes a median of 849 ms per query. Second, oracle calls are issued in sequential rounds: the shaping sample, the refinement rounds (median 6, 90th percentile 8), then the oracle region and the certification draws in parallel, then gap closing in batches of 50. Gap closing is not needed in 54% of queries; the median number of gap-closing rounds is at most 4 on any dataset, and the 90th percentile at most 41. Each round adds the latency of one batch of parallel oracle calls, the main latency cost of CERTIFIEDFILTER relative to a single-round cascade; fewer, larger rounds would trade calls for latency.

8 Discussion and Lessons

What limits the savings? Two factors bound the savings: the certification floor, and how well past answers predict new ones. To measure the floor of Proposition 4.2, we replay every workload with a *perfect* score, the oracle label itself, while certification samples as usual. SEMVIEWS then still needs 31% of the calls of exact evaluation on average, against 60% with its learned score: 80% on predicates with at most 2% selectivity and 14% on those above 10%. About half of the remaining cost is thus a property of our 5,000-tuple relations, not of the score. Because the floor does not grow with the relation, its share shrinks on larger tables. To vary only the size, we simulate larger relations by repeating each relation’s tuples 4 and 20 times, which preserves every label, score, and embedding and each predicate’s selectivity. The share of exact evaluation falls from 60% to 52% at 20,000 tuples and 51% at 100,000 (48% with a shaping sample of 0.8% of the table), with every miss rate below δ , while the perfect-score cost drops to 7%. The simulation cannot tell whether real large tables are easier or harder to score, but it shows where the limit moves: on large relations the floor nearly vanishes and almost all of the remaining cost is the score’s uncertainty, which refinement reduces and related views can reduce further.

The second factor is the predictive power of past answers. Even with every other predicate’s labels known exactly, a logistic model separates the predicates of a pool with a median AUROC of 0.87 on PubMed, 0.82 on Reviews, 0.95 on GoEmotions, and 0.95 on DBpedia. Real predicates share many positives and negatives, but rarely so cleanly that a past answer settles a new one; this is precisely why views must inform the order and not the answer.

When do views help, and how general is the gain? Views help substantially when the workload contains a related predicate and cost almost nothing otherwise (Sections 7.3 and 7.6), so how much a whole workload gains is a property of the workload, not of the operator. Our pools are a mixed test. GoEmotions, where views help most, is generated from a taxonomy whose group predicates (Ekman groups, sentiments) are near-unions of other predicates in the pool, and it was one of the two pilot datasets used to fix hyperparameters; its gain is likely an upper end. PubMed’s human-written external pool has few related predicates and gains little. Even a perfectly warm catalog, holding the complete oracle labels of every other predicate before each query, saves only 5%–16% more calls than the workloads we replay, most on GoEmotions: on these pools, there is little more for views to give. Analyst and agent sessions that refine a question step by step may hold more related predicates than a random order of a pool, but we have not measured

such sessions. The gains of the certifier and of free row embeddings over per-query baselines, in contrast, appear on all four datasets.

Limitations. Our main relations have 5,000 tuples; the tiling simulation and larger-scale experiments (Appendix B) show savings growing with the relation, but we have not measured relations of millions of tuples. Each workload issues every predicate of a pool in a random order, and two pools are generated from taxonomies; real analyst and agent sessions and longer-lived catalogs remain to be measured. The two datasets used to fix hyperparameters, GoEmotions and PubMed, are not fully held out (Appendix C). Finally, we study filters only.

9 Related Work

Semantic operator systems. LOTUS [38, 39] defines semantic operators with a reference-algorithm semantics, which we adopt. Palimpsest [31] and Abacus [42] choose models, prompts, and implementations under cost and quality constraints; DocETL [46] rewrites document pipelines with an agent; UQE [7] combines sampling and LLM calls for unstructured queries; Nirvana [58], CADENZA [14], and PLOP [37] optimize the logical and physical plan of one query; SEMA-SQL [27] generates and optimizes hybrid queries; and Cortex AISQL [28] brings AI-aware optimization and cascades into a production SQL engine. QUEST [49] reduces the tokens of each extraction call. These systems optimize one query at a time and do not use the oracle’s labels for one predicate to answer another. SEMVIEWS is a plug-in for such systems: it changes how one filter is executed and keeps the labels for later queries.

Per-query optimization with guarantees. The LOTUS cascade [39] learns thresholds on a helper score for precision and recall targets with normal-approximation bounds adapted from SUPG [20]; BARGAIN [54] replaces them with betting-based bounds [52] and adaptive sampling, and streaming cascades [29] refine thresholds as labels arrive. ScaleDoc [56], CSV [18], Larch [57], Task Cascades [47], lightweight proxies [5], and a unified template for semantic filtering [23] cut the cost of one filter further, and FDJ [55] prunes semantic joins with guarantees. Query-agnostic embedding indexes such as TASTI [22] are reused across queries, as are thresholds and proxies for the *same* predicate in LOTUS’s CascadeOptimizer [1] and in offline proxies [5]. Stretto [43] and ConRAD [17] allocate error budgets across the operators of one plan. All of these pay for every label of the current predicate. SEMVIEWS is complementary: it shrinks the set on which a per-query method has to work.

Reuse across predicates and queries. SemWeave [36] is closest to our goal. It intercepts semantic filters in Palimpsest and LOTUS, asks an LLM for the entailment relation between a new filter and similar past ones, and derives tuples’ labels from stored results when the relation determines them, treating a prior result as a materialized semantic view. It checks relations for logical consistency but offers no guarantee on the result. EVA [53] reuses the outputs of the same expensive UDF across exploratory queries whose predicates overlap symbolically. Both trust the inferred relation; we show that judged relations often fail on the oracle’s labels and that statistical containment degrades along chains, and SEMVIEWS certifies every reuse instead. Exact-match caches [10, 24] and memoization layers [30] hit only on an identical prompt and tuple; Sema [40]

deduces relational constraints from a single predicate. Similarity caches [12, 41, 59] reuse answers to nearby prompts without guarantees, and work on their eviction policies [34] is relevant to view retention. vCache [44] bounds the per-response error of such a cache, not the precision and recall of a result set. Semantic caching for OLAP [3] reuses results through exact intent matches and logical derivations over structured queries. SemBaker [9] compiles semantic operators into generated code that replaces the oracle, a different trade-off from reusing oracle labels.

Views, caching, and multi-query optimization. Answering queries using views [15], view matching in optimizers [13], view selection [16], semantic data caching [8], and multi-query optimization [45] all rely on logical containment between predicates over structured data. Semantic predicates are related only statistically, which is why SEMVIEWS uses views to plan but not to decide. Probabilistic predicates [35] train cheap filters ahead of time for one predicate, and semantic integrity constraints [26] declare and check constraints on semantic operator outputs; logical relations between predicates could be stated as such constraints but would still need certification.

Sampling, estimation, and statistical inference. Stratified sampling for approximate query processing [4] chooses strata from the workload to answer aggregates; ABae [21] stratifies by a proxy score for aggregates with expensive predicates; OLLA [19] and query-centric AI workflows [51] aggregate over LLM outputs online with confidence intervals. Prediction-powered inference [2], its stratified variant [11], and active statistical inference [61] combine model predictions with labeled samples for valid estimates. Semantic histograms [50] and the SemCEB benchmark [60] estimate the selectivity of semantic predicates. These methods estimate aggregates or selectivities; SEMVIEWS certifies the precision and recall of a set-valued answer. Benchmarks of semantic queries such as SemBench [25] could supply further workloads.

10 Conclusion

SEMVIEWS shows that a semantic filter can be both certified and cheap. By ordering tuples with side information that costs no oracle calls, refining the order with labels it must pay for anyway, and certifying only with fresh samples, it saves several times as many calls as the strongest prior per-query method with the same guarantee. The same separation makes the oracle’s labels safely reusable across queries: views help in proportion to how related a workload’s predicates are, most when a question returns in other words, whereas uncertified reuse by similarity or entailment misses its targets on a large share of queries. More broadly, letting side information shape the plan while only fresh samples enter a bound lets a semantic engine exploit any knowledge it has accumulated without risking its guarantees. Extending certified reuse to semantic joins and ranking, and measuring how often real sessions revisit their questions, remain open.

References

- [1] 2026. LOTUS (lotus-ai), version 1.2.4. Python package. <https://pypi.org/project/lotus-ai/> LazyFrame API reference: https://lotus-ai.readthedocs.io/en/stable/lazyframe_api.html.

- [2] Anastasios N. Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I. Jordan, and Tijana Zrnica. 2023. Prediction-Powered Inference. arXiv:2301.09633 <https://arxiv.org/abs/2301.09633>
- [3] Laurent Bindschaedler. 2026. Semantic Caching for OLAP via LLM-Based Query Canonicalization (Extended Version). arXiv:2602.19811 <https://arxiv.org/abs/2602.19811>
- [4] S. Chaudhuri, Gautam Das, and Vivek R. Narasayya. 2007. Optimized stratified sampling for approximate query processing. *ACM Transactions on Database Systems* 32 (2007), 9. <https://doi.org/10.1145/1242524.1242526>
- [5] Yeounoh Chung, Rushabh Desai, Jian He, Yu Xiao, Thibaud Hottelier, Yves-Laurent Kom Samo, Pushkar Khadilkar, Xianshun Chen, Sam Idicula, Fatma Özcan, Alon Halevy, and Yannis Papakonstantinou. 2026. 100x Cost & Latency Reduction: Performance Analysis of AI Query Approximation using Lightweight Proxy Models. In *Proceedings of the 2026 ACM SIGMOD International Conference on Management of Data (SIGMOD '26)*. <https://doi.org/10.1145/3802002>
- [6] C. J. Clopper and E. S. Pearson. 1934. The Use of Confidence or Fiducial Limits Illustrated in the Case of the Binomial. *Biometrika* 26 (1934), 404–413. <https://doi.org/10.1093/biomet/26.4.404>
- [7] Hanjun Dai, Bethany Yixin Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Phitchaya Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans. 2024. UQE: A Query Engine for Unstructured Databases. In *NeurIPS*. <https://arxiv.org/abs/2407.09522>
- [8] Shaul Dar, Michael J. Franklin, Björn T. Jonsson, Divesh Srivastava, and Michael Tan. 1996. Semantic Data Caching and Replacement. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB)*. <https://www.vldb.org/conf/1996/P330.PDF>
- [9] Wenkai Dong and Yifan Wang. 2026. From Interpretation to Compilation: A Compilation-Based Execution Engine for Semantic Operator Systems. arXiv:2608.06677 <https://arxiv.org/abs/2608.06677>
- [10] Anas Dorbani, Sunny Yasser, Jimmy Lin, and Amine Mhedhbi. 2025. Beyond Quacking: Deep Integration of Language Models and RAG into DuckDB. *Proc. VLDB Endow.* 18, 12 (2025), 5415–5418. <https://doi.org/10.14778/3750601.3750685>
- [11] Adam Fisch, Joshua Maynez, R. Alex Hofer, Bhuvan Dhingra, Amir Globerson, and William W. Cohen. 2024. Stratified Prediction-Powered Inference for Hybrid Language Model Evaluation. arXiv:2406.04291 <https://arxiv.org/abs/2406.04291>
- [12] Waris Gill, Mohamed Elidrisi, Pallavi Kalapatapu, Ammar Ahmed, Ali Anwar, and Muhammad Ali Gulzar. 2025. MeanCache: User-Centric Semantic Caching for LLM Web Services. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. <https://arxiv.org/abs/2403.02694>
- [13] Jonathan Goldstein and P. Larson. 2001. Optimizing queries using materialized views: a practical, scalable solution. In *ACM SIGMOD Conference*. 331–342. <https://doi.org/10.1145/375663.375706>
- [14] Jaehyun Ha, Yongjoo Park, and Wook-Shin Han. 2026. CADENZA: Compiling Natural-Language Intent into Task-Specific Operator DAGs for Semantic Query Processing. arXiv:2606.29151 <https://arxiv.org/abs/2606.29151> Accepted to SIGMOD 2027 (arXiv comment).
- [15] A. Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal* 10 (2001), 270–294. <https://doi.org/10.1007/s007780100054>
- [16] Venky Harinarayan, A. Rajaraman, and J. Ullman. 1996. Implementing data cubes efficiently. In *ACM SIGMOD Conference*. 205–216. <https://doi.org/10.1145/233269.233333>
- [17] Sonia Horchidan, Fabian Zeiher, Xiangyu Shi, Vasiliki Kalavri, Henrik Boström, Ioannis Kontoyiannis, and Paris Carbone. 2026. ConRAD: Conformal Risk-Aware Neural Databases. arXiv:2605.03806 <https://arxiv.org/abs/2605.03806>
- [18] Nan Hou, Kangfei Zhao, Jiadong Xie, and Jeffrey Xu Yu. 2026. Beyond Linear LLM Invocation: An Efficient and Effective Semantic Filter Paradigm. arXiv:2603.04799 <https://arxiv.org/abs/2603.04799>
- [19] Chao Hui, Weizheng Lu, Yanjie Gao, Lingfeng Xiong, Yunhai Wang, and Yueguo Chen. 2026. LLM-Driven Online Aggregation for Unstructured Text Analytics. arXiv:2603.08443 <https://arxiv.org/abs/2603.08443> DASFAA 2026 (arXiv comment).
- [20] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate Selection with Guarantees using Proxies. *Proc. VLDB Endow.* 13, 11 (2020), 1990–2003. <https://doi.org/10.14778/3407790.3407804>
- [21] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating Approximate Aggregation Queries with Expensive Predicates. *Proc. VLDB Endow.* 14, 11 (2021), 2341–2354. <https://doi.org/10.14778/3476249.3476285>
- [22] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. TASTI: Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD)*. <https://doi.org/10.1145/3514221.3517897>
- [23] Kyoungmin Kim, Martin Catheland, and Anastasia Ailamaki. 2026. Fast LLM-Based Semantic Filtering: From a Unified Framework to an Adaptive Two-Phase Method. arXiv:2606.08090 <https://arxiv.org/abs/2606.08090>
- [24] Udes Kumarasinghe, Tyler Liu, Ahmed R. Mahmood, Chunwei Liu, and Walid G. Aref. 2026. iPDB – Optimizing Semantic SQL Queries. arXiv:2601.16432 <https://arxiv.org/abs/2601.16432>
- [25] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. 2025. SemBench: A Benchmark for Semantic Query Processing Engines. arXiv:2511.01716 <https://arxiv.org/abs/2511.01716> Accepted to VLDB 2026 (arXiv comment). Code: <https://github.com/SemBench/SemBench>
- [26] Alexander W. Lee, Justin Chan, Michael Fu, Nicolas Kim, Akshay Mehta, Deepti Raghavan, and Ugur Cetintemel. 2025. Semantic Integrity Constraints: Declarative Guardrails for AI-Augmented Data Processing Systems. *Proc. VLDB Endow.* 18, 11 (2025), 4073–4080. <https://www.vldb.org/pvldb/vol18/p4073-lee.pdf>
- [27] Yin Lin, Tianjing Zeng, Zhongjun Ding, Rong Zhu, Bolin Ding, H. V. Jagadish, and Jingren Zhou. 2026. SEMA-SQL: Beyond Traditional Relational Querying with Large Language Models. arXiv:2604.23477 <https://arxiv.org/abs/2604.23477>
- [28] Paweł Liskowski, Benjamin Han, Paritosh Aggarwal, Bowei Chen, Boxin Jiang, Nitish Jindal, Zihan Li, Aaron Lin, Kyle Schmaus, Jay Tayade, Weicheng Zhao, Anupam Datta, Nathan Wiegand, and Dimitris Tsrogiannis. 2026. Cortex AISQL: A Production SQL Engine for Unstructured Data. In *Companion of the International Conference on Management of Data (SIGMOD Companion), Industry Track*. <https://doi.org/10.1145/3788853.3803093>
- [29] Paweł Liskowski and Kyle Schmaus. 2026. Streaming Model Cascades for Semantic SQL. arXiv:2604.00660 <https://arxiv.org/abs/2604.00660>
- [30] Paweł Liskowski, Fuheng Zhao, Benjamin Han, Anupam Datta, and Dimitris Tsrogiannis. 2026. Compositional Online Learning for Semantic Data Processing Systems. arXiv:2608.27244 <https://arxiv.org/abs/2608.27244>
- [31] Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baille Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, and Gerardo Vitagliano. 2024. A Declarative System for Optimizing AI Workloads. arXiv:2405.14696 <https://arxiv.org/abs/2405.14696>
- [32] Shu Liu, Asim Biswal, Amog Kamsetty, Audrey Cheng, Luis Gaspar Schroeder, Liana Patel, Shiyi Cao, Xiangxi Mo, Ion Stoica, Joseph E. Gonzalez, and Matei Zaharia. 2024. Optimizing LLM Queries in Relational Data Analytics Workloads. arXiv:2403.05821 <https://arxiv.org/abs/2403.05821>
- [33] Shu Liu, Soujanya Ponnappalli, Shreya Shankar, Sepanta Zeighami, Alan Zhu, Shubham Agarwal, Ruiqi Chen, Samion Suwito, Shuo Yuan, Ion Stoica, Matei Zaharia, Alvin Cheung, Natacha Crooks, Joseph E. Gonzalez, and Aditya G. Parameswaran. 2026. Supporting Our AI Overlords: Redesigning Data Systems to be Agent-First. In *Conference on Innovative Data Systems Research (CIDR)*. <https://www.vldb.org/cidrdb/2026/supporting-our-ai-overlords-redesigning-data-systems-to-be-agent-first.html>
- [34] Xutong Liu, Baran Atalar, Xiangxiang Dai, Jinhang Zuo, Siwei Wang, John C. S. Lui, Wei Chen, and Carlee Joe-Wong. 2025. Semantic Caching for Low-Cost LLM Serving: From Offline Learning to Online Adaptation. arXiv:2508.07675 <https://arxiv.org/abs/2508.07675> Accepted to INFOCOM 2026 (arXiv comment).
- [35] Yao Lu, A. Chowdhery, Srikanth Kandula, and S. Chaudhuri. 2018. Accelerating Machine Learning Inference with Probabilistic Predicates. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*. <https://doi.org/10.1145/3183713.3183751>
- [36] Md. Tareq Mahmood, Venkatesh Emani, Hangdong Zhao, and Shivaram Venkataraman. 2026. Demo of SemWeave: Semantic Common Expressions for LLM-powered Query Processing. In *Companion of the International Conference on Management of Data (SIGMOD Companion)*. 102–105. <https://doi.org/10.1145/3788853.3801593>
- [37] Qiuyang Mang, Yufan Xiang, Hangrui Zhou, Runyuan He, Jiaxiang Yu, Hanchen Li, Aditya Parameswaran, and Alvin Cheung. 2026. PLOP: Cost-Based Placement of Semantic Operators in Hybrid Query Plans. arXiv:2604.09944 <https://arxiv.org/abs/2604.09944>
- [38] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2024. Semantic Operators: A Declarative Model for Rich, AI-based Data Processing. arXiv:2407.11418 [cs.DB] <https://arxiv.org/abs/2407.11418>
- [39] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Enabling LLM-Based Data Processing with Accuracy Guarantees in LOTUS. *Proc. VLDB Endow.* 18, 11 (2025), 4171–4184. <https://doi.org/10.14778/3749646.3749685>
- [40] Kangkang Qi, Dongyang Xie, Wenbo Li, Hao Zhang, Yuanyuan Zhu, Jeffrey Xu Yu, and Kangfei Zhao. 2026. Sema: A High-performance System for LLM-based Semantic Query Processing. arXiv:2603.11622 <https://arxiv.org/abs/2603.11622>
- [41] Sajal Regmi and Chetan Phakami Pun. 2024. GPT Semantic Cache: Reducing LLM Costs and Latency via Semantic Embedding Caching. arXiv:2411.05276 <https://arxiv.org/abs/2411.05276>
- [42] Matthew Russo, Chunwei Liu, Sivaprasad Sudhir, Gerardo Vitagliano, Michael Cafarella, Tim Kraska, and Samuel Madden. 2026. Abacus: A Cost-Based Optimizer for Semantic Operator Systems. *Proc. VLDB Endow.* 19, 5 (2026), 1060–1073. <https://doi.org/10.14778/3796195.3796215>
- [43] Gabriele Sammartino, Matthias Urban, Paolo Papotti, and Carsten Binnig. 2026. The Stretto Execution Engine for LLM-Augmented Data Systems. arXiv:2602.04430 <https://arxiv.org/abs/2602.04430>

- [44] Luis Gaspar Schroeder, Aditya Desai, Alejandro Cuadron, Kyle Chu, Shu Liu, Mark Zhao, Stephan Krusche, Alfons Kemper, Matei Zaharia, and Joseph E. Gonzalez. 2025. vCache: Verified Semantic Prompt Caching. arXiv:2502.03771 <https://arxiv.org/abs/2502.03771> ICLR 2026 (accepted), per arXiv comment.
- [45] T. Sellis. 1988. Multiple-query optimization. *ACM Transactions on Database Systems* 13 (1988), 23–52. <https://doi.org/10.1145/42201.42203>
- [46] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2024. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. arXiv:2410.12189 <https://arxiv.org/abs/2410.12189>
- [47] Shreya Shankar, Sepanta Zeighami, and Aditya Parameswaran. 2026. Task Cascades for Efficient Unstructured Data Processing. arXiv:2601.05536 <https://arxiv.org/abs/2601.05536> SIGMOD 2026 (arXiv comment).
- [48] Hojae Son, Md Ashrafur Islam, Huy Gia Cao, Hui Guan, and Marco Serafini. 2026. Kalypso: Relational LLM Serving. arXiv:2607.23815 <https://arxiv.org/abs/2607.23815>
- [49] Zhaoze Sun, Qiyang Deng, Chengliang Chai, Kaisen Jin, Xinyu Guo, Han Han, Ye Yuan, Guoren Wang, and Lei Cao. 2025. QUEST: Query Optimization in Unstructured Document Analysis. arXiv:2507.06515 <https://arxiv.org/abs/2507.06515> Accepted by VLDB 2025 (arXiv comment).
- [50] Matthias Urban, Vu Huy Nguyen, Gabriele Sanmartino, Paolo Papotti, and Carsten Binnig. 2026. Selectivity Estimation for Semantic Filters on Image Data. In *VLDB 2026 Workshop: Applied AI for Database Systems and Applications (AIDB 2026)*. https://www.vldb.org/2026/Workshops/VLDB-Workshops-2026/AIDB/aidb26_5.pdf
- [51] Huayi Wang, Jun Xu, and Gromit Yeuk-Yin Chan. 2026. Query-Centric Optimization of AI Workflows via Approximate Query Processing and Proxy Models. arXiv:2607.00254 <https://arxiv.org/abs/2607.00254>
- [52] Ian Waudby-Smith and Aaditya Ramdas. 2020. Estimating means of bounded random variables by betting. arXiv:2010.09686 <https://arxiv.org/abs/2010.09686>
- [53] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Kishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. <https://doi.org/10.1145/3514221.3526142>
- [54] Sepanta Zeighami, Shreya Shankar, and Aditya Parameswaran. 2025. Cut Costs, Not Accuracy: LLM-Powered Data Processing with Guarantees. arXiv:2509.02896 <https://arxiv.org/abs/2509.02896> Forthcoming in SIGMOD 2026 (arXiv comment).
- [55] Sepanta Zeighami, Shreya Shankar, and Aditya Parameswaran. 2025. Featurized-Decomposition Join: Low-Cost Semantic Joins with Guarantees. arXiv:2512.05399 <https://arxiv.org/abs/2512.05399>
- [56] Hengrui Zhang, Yulong Hui, Yihao Liu, and Huanchen Zhang. 2025. ScaleDoc: Scaling LLM-based Predicates over Large Document Collections. <https://doi.org/10.1145/3802106> arXiv:2509.12610
- [57] Fuheng Zhao, Pawel Liskowski, Zihan Li, Benjamin Han, Puxuan Yu, Varich Boonsanong, Dimitris Tsirogiannis, and Anupam Datta. 2026. Larch: Learned Query Optimization for Semantic Predicates. arXiv:2606.07923 <https://arxiv.org/abs/2606.07923>
- [58] Junhao Zhu, Lu Chen, Xiangyu Ke, Ziquan Fang, Tianyi Li, Yunjun Gao, and Christian S. Jensen. 2025. Beyond Relational: Semantic-Aware Multi-Modal Analytics with LLM-Native Query Optimization. arXiv:2511.19830 <https://arxiv.org/abs/2511.19830>
- [59] Zilliz. 2023. GPTCache: Semantic cache for LLMs. GitHub repository. <https://github.com/zilliztech/GPTCache>
- [60] Andreas Zimmerer, Claudius Kühn, Yang Li, Mihail Stoian, Renata Borovica-Gajic, and Andreas Kipf. 2026. SemCEB: A Cardinality Estimation Benchmark for Semantic Operators. arXiv:2606.23081 <https://arxiv.org/abs/2606.23081>
- [61] Tijana Zrnica and Emmanuel J. Candès. 2024. Active Statistical Inference. arXiv:2403.03208 <https://arxiv.org/abs/2403.03208>

A Proofs

A.1 Proof of Proposition 3.2

Let $P, Q, \mathcal{R}$ be the sets of tuples on which the oracle labels p, q, r positive, with $|P| > 0, |Q| > 0$. Assume $|P \setminus Q| \leq \varepsilon_1 |P|$ and $|Q \setminus \mathcal{R}| \leq \varepsilon_2 |Q|$. Then

$$|P \setminus \mathcal{R}| = |(P \setminus Q) \setminus \mathcal{R}| + |(P \cap Q) \setminus \mathcal{R}| \leq |P \setminus Q| + |Q \setminus \mathcal{R}| \leq \varepsilon_1 |P| + \varepsilon_2 |Q|,$$

and dividing by $|P|$ gives $\Pr[r=0 \mid p=1] \leq \varepsilon_1 + \varepsilon_2 \text{sel}(q)/\text{sel}(p)$.

Tightness. Fix $|P|, |Q|, \varepsilon_1, \varepsilon_2$ with $\varepsilon_2 |Q| \leq (1 - \varepsilon_1) |P|$ (the condition of the proposition; $|Q| \geq (1 - \varepsilon_1) |P|$ then holds because $Q \supseteq P \cap Q$). Choose $P \setminus Q$ of size $\varepsilon_1 |P|$, choose $Q \supseteq P \cap Q$ with $|P \cap Q| = (1 - \varepsilon_1) |P|$, and let $\mathcal{R}$ exclude exactly $P \setminus Q$ and every $Q \setminus P$ tuple of $P \cap Q$ (possible by the first condition), and include any other tuple. Then $p \sqsubseteq_{\varepsilon_1} q$

and $q \sqsubseteq_{\varepsilon_2} r$ hold with equality, and $|P \setminus \mathcal{R}| = \varepsilon_1 |P| + \varepsilon_2 |Q|$, so both inequalities above are equalities.

A.2 Proof of Theorem 4.1

Setting. Fix the relation, the oracle labels, the catalog, the helper scores, and the embeddings. Let $\mathcal{F}$ denote everything CERTIFIED-FILTER observes before its certification draws: the catalog, the shaping sample, the refinement tuples and their labels, and the fitted score $\hat{\pi}$. Refinement is part of $\mathcal{F}$: it labels tuples, refits, and re-plans, but all of it happens before the regions are fixed and the certification draws are made. Conditional on $\mathcal{F}$, the regions X, A, O and the sample sizes n_X, n_A are fixed, and so is the split of δ ($\alpha_X + \alpha_A = \delta$, or δ when only one region is non-empty). The certification draws are made after $\mathcal{F}$ with fresh randomness: n_X tuples uniformly without replacement from X and n_A from A . Conditional on $\mathcal{F}$, the number of positives among the draws from X is hypergeometric with parameters $(|X|, c_X, n_X)$, where c_X is the (fixed) number of oracle positives in X ; likewise for A .

Coverage event. Let $u_X = \max\{c : \Pr_{\text{HG}(|X|, c, n_X)}[x \leq x_X] > \alpha_X\}$ and $\ell_A = \min\{c : \Pr_{\text{HG}(|A|, c, n_A)}[x \geq x_A] > \alpha_A\}$ be the one-sided exact hypergeometric bounds computed from the numbers of positives x_X and x_A among the draws. By the standard test-inversion argument (the tail probabilities are monotone in c), conditional on $\mathcal{F}$, $\Pr[c_X > u_X] \leq \alpha_X$ and $\Pr[c_A < \ell_A] \leq \alpha_A$. Let $E = \{c_X \leq u_X\} \cap \{c_A \geq \ell_A\}$. A union bound gives $\Pr[E^c \mid \mathcal{F}] \leq \delta$, hence $\Pr[E^c] \leq \delta$ unconditionally. No other quantity computed later enters a bound.

Deterministic step. Gap closing only adds oracle labels of tuples in X or A ; it draws nothing and recomputes no bound. Consider any state the algorithm can stop in. Let P be the number of known positives outside A (in K , which includes the shaping and refinement labels, in O , and those found in X), d_X and v_X the known positives and the still-unknown tuples in X , d_A and e_A the known positives and negatives in A . The returned set is all known positives plus A without its known negatives. Its true positives are $P + c_A$ (no positive of A is dropped), its size is $P + |A| - e_A$, and the positives it misses are the unknown positives of X , numbering $c_X - d_X \leq v_X$. Therefore

$$\text{rec} = \frac{P + c_A}{P + c_A + (c_X - d_X)}, \quad \text{prec} = \frac{P + c_A}{P + |A| - e_A}.$$

Both are non-decreasing in c_A , and the recall is non-increasing in c_X . On E , $c_A \geq \max(\ell_A, d_A)$ and $c_X - d_X \leq \min(\max(u_X - d_X, 0), v_X)$; substituting yields exactly the two quantities that the stopping rule requires to reach γ_R and γ_P . So on E , every state in which the algorithm stops meets both targets, and the algorithm always stops (labeling X and A completely sets $v_X = 0$ and drops every negative of A , so both quantities equal 1). Hence $\Pr[\text{targets met}] \geq \Pr[E] \geq 1 - \delta$.

What the proof does not use. It uses no property of the views, the helper, the embeddings, or the fitted score; they only determine X, A, O, n_X, n_A through $\mathcal{F}$. The shaping labels are known exactly and contribute only to P , and so do the refinement labels. Sampling without replacement makes the draws exactly hypergeometric for the finite region, so no approximation is involved; with replacement and Clopper–Pearson bounds, as in the ablation “first draft”, the

same argument holds with binomial draws. The guarantee is per query: for each query in a workload, conditional on everything that happened before it, the probability of missing a target is at most δ . The miss indicators of a workload therefore form a sequence whose conditional means are at most δ , so by the Azuma–Hoeffding inequality the miss rate of a long workload exceeds δ by more than $O(1/\sqrt{n})$ only with small probability; the validity experiment of Section 7.4 measures this rate.

General strata. The same argument extends to m predefined strata decided individually as accept, reject, or oracle, with simultaneous bounds from a union bound over $2m$ one-sided bounds (and over a predefined schedule of L looks, at level $\delta w_i/(2L)$ per bound with $\sum_i w_i \leq 1$): the recall and precision ratios are monotone in every stratum’s positive count, so plugging in the bounds is valid for every assignment simultaneously. The ablation “per-signature strata” in Section 7.7 is this variant.

B Larger-Scale Experiments

Our main relations have 5,000 tuples, so certification samples are a large part of each query (Section 8). We chose that size to limit the cost and carbon footprint of the experiments: measuring validity requires the oracle’s label for every tuple and predicate, so the labeling cost grows with the number of tuples times the number of predicates. To verify that our findings hold on larger relations without labeling every predicate, we built two real relations of 100,000 tuples from the sources of the main tables, disjoint from them: DBpedia articles from the training split (the 5,000-tuple table comes from the test split), and Amazon Electronics reviews from the start of the same source file, without the reviews of the 5,000-tuple table. For each relation we chose 8 predicates of its pool at random with a fixed seed, had the oracle label every tuple (1,600,000 labels in total; selectivity 0.9%–19.6% on DBpedia and 0.4%–75.0% on Reviews), and replayed 20 random orders of each pool. On DBpedia we also ran all 40 predicates with the human ontology labels as the oracle (4 orders). That oracle is exact and free but is not the LLM: on the 5,000-tuple table the two agree on 97.7% of tuples (median over predicates) with a median F1 of only 0.65, so we use it only as a second, larger workload. The first 5,000 and 20,000 tuples of each relation, with the same labels, serve as the smaller tables, so within each block of Table 10 only the table size changes. The operator is the paper’s with two changes: the helper is off, because scoring every tuple–predicate pair with it locally would take more than a day (without it, SEMVIEWS needs no more calls on the 5,000-tuple DBpedia and Reviews tables; Table 9), and above 5,000 tuples the certification sample may be as large as in the tiling simulation. Nothing was tuned on these data.

Results. SEMVIEWS stays valid at every size: the share of queries that miss a target never exceeds 1.2% on DBpedia and 0.6% on Reviews with the LLM oracle, and 0.6% with human labels, against $\delta = 5\%$. Its relative cost falls as the table grows: from 60% of exact evaluation at 5,000 tuples to 34% at 100,000 on DBpedia, from 77% to 55% on Reviews, and from 52% to 20% with human labels. Rare predicates, whose cost on small tables is mostly the certification floor, gain most: on Reviews their share drops from 95% to 66%. The real tables thus confirm the trend of the tiling simulation, which

Table 10: SEMVIEWS on relations of up to 100,000 tuples: oracle calls as a share of exact evaluation (pooled over queries and orders), for all predicates and for those with selectivity at most 2%, and the share of queries meeting both targets ($p = r = 0.9$, $\delta = 0.05$). Smaller tables are prefixes of the same relation. Helper off.

Oracle	Tuples	Share of exact		Rare ($\leq 2\%$)	Both met
		SEMVIEWS	no views		
DBpedia, human labels (40 predicates)	5,000	52%	58%	64%	100.0%
	20,000	31%	35%	40%	100.0%
	100,000	20%	21%	23%	99.4%
DBpedia, LLM oracle (8 predicates)	5,000	60%	60%	67%	100.0%
	20,000	45%	46%	68%	100.0%
	100,000	34%	34%	34%	99.4%
Reviews, LLM oracle (8 predicates)	5,000	77%	77%	95%	99.4%
	20,000	64%	63%	76%	100.0%
	100,000	55%	54%	66%	100.0%

predicted 51% at 100,000 tuples on average over the four datasets; DBpedia falls below that and Reviews stays above it. Views add nothing here: predicates drawn at random from a pool of 40 are rarely related, as in the unrelated control of Section 7.6, so the gain at scale comes from the certifier and the learned score.

C Reproducibility

All results replay recorded oracle labels; the scripts, configurations, and recorded labels needed to regenerate every figure and table are at <https://github.com/semviews/lotus-semviews>. Hyperparameters (shaping-sample size 200, regularization, number of candidate views, region grid) were fixed on synthetic data and on partial label records of GoEmotions and PubMed before the full runs, and frozen in the experiment configuration. The refinement rounds, their batch, the planning margin, the δ split, and sampling without replacement were then chosen in three tuning rounds on GoEmotions and PubMed only, each round on fresh workload orders (orders that the reported runs on these two datasets also include), with a check on GoEmotions and PubMed tiled 4 and 20 times that led to a batch proportional to the relation. The two tuning datasets are therefore not fully held out; DBpedia and Reviews are. The ablation reports the first-draft operator, without refinement and with Clopper–Pearson bounds, for comparison.

D AI-use disclosure

Generative AI tools were used to write parts of the code and text under the authors’ direction; the details required by the ACM policy are listed in the repository’s docs/AI_USE.md.