Efficient and Robust Evaluation of Large-Scale AI Agents Using Coresets

AI Systems Group
contact@system-design.ai

November 10, 2025

Abstract

As generative AI agents and chatbots are deployed at scale, they produce a deluge of user interactions, trajectories, and utterances. Evaluating performance, tracking regressions, and identifying drift are operationally critical yet expensive under naive sampling. We present a practical, theoretically grounded framework for applying *coresets*—small, weighted subsets that provably approximate a larger set—to agent evaluation. We formalize objectives, give construction strategies (geometric coverage, sensitivity-based importance sampling, error-focused active coresets), and derive weighting and variance-control schemes that yield unbiased or low-bias, low-variance estimates of key metrics across slices. We provide streaming algorithms with merge-and-reduce guarantees and distributional drift tests (MMD, Wasserstein, Fréchet-type distances) tied to alerting. We close with validation protocols, limits, and open problems for multi-objective preservation (e.g., F1, P_{99} latency, and calibration) under adversarial shifts.

1 The Evaluation Bottleneck in Deployed AI

Deployed agents can generate billions of utterances and millions of multi-turn trajectories weekly. Teams must answer:

- Is model v_{n+1} better than v_n?
- Did a critical task (e.g., "book a flight") regress?
- Are jailbreaks or subtle failures emerging in a user segment?
- Are SLOs (latency, tool-call success, cost) holding under load?

Exhaustive labeling is intractable; uniform subsampling misses rare yet business-critical behaviors. A *coreset* S is a small, weighted subset approximating a large dataset X for a class of objectives (Feldman & Langberg, 2011; Braverman et al., 2016; Lucic et al., 2018). Properly constructed, S supports high-fidelity metric estimation, reproducible regression tests, and early drift detection.

Setting. Let $X = \{(x_i, w_i)\}_{i=1}^N$ denote trajectories with traffic weights $w_i \geq 0$ (e.g., $w_i = 1/N$ or empirical frequencies). For a metric f (accuracy, F1, reward, cost), define

$$\hat{f}_X = \sum_{i=1}^{N} w_i \, f(x_i). \tag{1}$$

A coreset $S = \{(s_j, v_j)\}_{j=1}^{k}$ with $k \ll N$ is ε-*accurate* for a function class $\mathcal{F}$ if

$$\sup_{f\in\mathcal{F}} \left|\hat{f}_X - \hat{f}_S\right| \;\le\; \varepsilon \quad \text{where} \quad \hat{f}_S \;=\; \sum_{j=1}^{k} v_j\, f(s_j). \tag{2}$$

Guarantees can be uniform (over $\mathcal{F}$), objective-specific (ERM loss), or distributional (IPM/MMD distances).

2 A Coreset Framework for Agent Evaluation

2.1 Step 1: Define the Evaluation Objective

Let $\mathcal{F}$ collect the metrics to preserve: outcome metrics (accuracy, F1, task success), operational metrics (latency quantiles, tool-call failure), calibration metrics (ECE, Brier), and slice constraints (compliance, VIP, high-revenue intents). For instance:

$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{N}\left|\text{acc}(b) - \text{conf}(b)\right|, \qquad \text{Brier} = \frac{1}{N}\sum_{i=1}^{N}\sum_{c}\left(p_{ic} - \mathbb{1}[y_i = c]\right)^2. \tag{3}$$

Define business thresholds ε_f so that $|\hat{f}_X - \hat{f}_S| \le \varepsilon_f$.

2.2 Step 2: Choose a Behavioral Representation

Construct $z_i \in \mathbb{R}^d$ for each x_i:

$$z_i = \left[z_{\text{embed}} \;\oplus\; z_{\text{meta}} \;\oplus\; z_{\text{difficulty}}\right],$$

where z_{embed} are sentence/trajectory embeddings (turn-level pooled), z_{meta} are structured features (tool graph counts, locale, device, outcome), and $z_{\text{difficulty}}$ includes judge-uncertainty, human–model disagreement, cost, policy flags. Normalize features; keep w_i alongside.

2.3 Step 3: Select a Construction Strategy

We detail four complementary strategies.

(A) Geometric coverage / diversity. *k-center greedy* (farthest-first) controls the maximum covering radius in Z (Gonzalez, 1985). *k-medoids/facility-location* maximizes a submodular representativeness objective $F(S) = \sum_{i\in X} w_i \max_{j\in S} \text{sim}(z_i, z_j)$ with $(1 - 1/e)$-approximation via greedy (Nemhauser et al., 1978). DPPs further encourage repulsion/diversity (Kulesza & Taskar, 2012).

(B) Sensitivity/importance sampling for ERM. For a proxy loss $L(\theta) = \sum_i w_i\, \ell(f_\theta(x_i), y_i)$, define point sensitivity

$$\sigma_i \;=\; \sup_{\theta\in\Theta} \frac{w_i\, \ell(f_\theta(x_i), y_i)}{\sum_j w_j\, \ell(f_\theta(x_j), y_j)}. \tag{4}$$

Sampling i with $p_i \propto \sigma_i w_i$ and reweighting $v_i \propto w_i/p_i$ yields (ε)-coresets whose size depends on $\sum_i \sigma_i$ but not N (Feldman & Langberg, 2011; Munteanu & Schwiegelshohn, 2018). For generalized linear models, leverage-score variants connect to ℓ_2-sensitivity (Drineas et al., 2012; Mahoney, 2011).

(C) Error-focused active coresets. Train a light failure predictor on cheap labels (judge model or heuristics). Select points by informativeness (e.g., BALD mutual information, margin sampling) with fairness-aware constraints across slices (Houlsby et al., 2011; Katharopoulos & Fleuret, 2018).

(D) Streaming merge-and-reduce. Process shards, build small coresets per shard, and recursively merge/reduce, preserving guarantees (Agarwal et al., 2012; Har-Peled & Mazumdar, 2018). This yields near-linear scalability and bounded memory.

2.4 Step 4: Agent-Specific Adaptations

Choose the evaluation unit as a *trajectory*. Include state-action signals: tool/API sequences, error codes (auth, schema, rate-limit), retries, function-call graphs. If SLOs matter, include latency percentiles or queueing proxies (e.g., service time vs. waiting time) in z_i or as explicit constraints.

2.5 Step 5: Weighting, Debiasing, Stratification

Let p_i denote the selection probability. The standard unbiased estimator for any metric f is the Horvitz–Thompson form

$$\hat{f}_S = \sum_{i \in S} \frac{w_i}{p_i} f(x_i), \qquad v_i \equiv \frac{w_i}{p_i}. \tag{5}$$

Its variance is

$$\text{Var}(\hat{f}_S) = \sum_i \frac{w_i^2}{p_i} \text{Var}\big(f(x_i)\big) + \sum_{i \neq j} \Big(\frac{w_i w_j}{p_i p_j} \text{Cov}(\mathbb{1}_{i \in S} f_i, \mathbb{1}_{j \in S} f_j) \Big), \tag{6}$$

which motivates p_i that scale with difficulty/variance (Neyman allocation) and negative dependence (e.g., DPP sampling) to reduce covariance.

Stratification. Partition $X = \bigsqcup_{g=1}^{G} X_g$ (intent, language, region, recency) and allocate k_g subject to $k = \sum_g k_g$. Within each stratum, run the chosen selector, compute p_i internally, and use (5). This prevents *rare-slice collapse*.

2.6 Step 6: Validation and Accept/Reject

Hold out a silent i.i.d. control set X_{holdout}. For each $f \in \mathcal{F}$:

$$\Delta_f = \big|\hat{f}_X - \hat{f}_S\big|, \quad \text{CI via normal or bootstrap on } \hat{f}_S, \tag{7}$$

$$\Delta_{f,g} = \big|\hat{f}_{X,g} - \hat{f}_{S,g}\big| \text{ for each slice } g, \tag{8}$$

$$\Delta_f^{\text{worst-}k} = \frac{1}{k} \sum_{g \in \text{worst-}k} \Delta_{f,g}. \tag{9}$$

Reject and revise if thresholds are exceeded; increase k, adjust strata or representation Z, or switch selector.

3 Theory Highlights and Useful Bounds

3.1 Uniform Approximation and Range Spaces

For range spaces with finite VC dimension d, ε-approximations of size $O(d\,\varepsilon^{-2}\log(d/\varepsilon))$ exist (Li et al., 2011). For clustering objectives (e.g., k-means), strong coreset sizes $O(dk\varepsilon^{-2})$ or better are known (Braverman et al., 2016; Feldman et al., 2013).

3.2 Sensitivity Sampling Guarantees

If $p_i \geq \min\{1, c\,\sigma_i/\sum_j \sigma_j\}$, then with $k = O((\sum_i \sigma_i)\,\varepsilon^{-2}\log(1/\delta))$ samples, the ERM loss $\sum_i w_i \ell(\cdot)$ is preserved within $(1 \pm \varepsilon)$ with probability $\geq 1-\delta$ (Feldman & Langberg, 2011; Munteanu & Schwiegelshohn, 2018). Practical proxies: gradient norms $\|\nabla_\theta \ell_i\|$, influence functions, or generalized leverage scores.

3.3 Variance Control and Concentration

For bounded $f \in [a,b]$ with independent sampling, Hoeffding gives

$$\mathbb{P}\big(|\hat{f}_S - \mathbb{E}[\hat{f}_S]| \geq t\big) \leq 2\exp\left(-\frac{2t^2}{\sum_{i\in S}(b-a)^2}\right).$$

Bernstein-type bounds incorporate variance, offering tighter CIs when f has heteroskedasticity. DPP sampling induces negative dependence, tightening tail bounds for linear statistics.

3.4 Bayesian Coresets (optional for reward/risk models)

For posterior approximations, Bayesian coresets (e.g., GIGA, Frank–Wolfe) greedily optimize a divergence objective to match the full-data log-likelihood geometry (Campbell & Broderick, 2018; Huggins et al., 2016). This supports fast posterior updates on small weighted subsets.

4 Drift Detection on Coreset Streams

Let S_t be the coreset for period t with embeddings Z_t and weights v_t. Distances between (Z_t, v_t) and (Z_{t-1}, v_{t-1}) trigger alerts.

MMD^2. For kernel k,

$$\mathrm{MMD}^2(P_t, P_{t-1}) \;=\; \mathbb{E}k(z,z') - 2\mathbb{E}k(z,\tilde{z}) + \mathbb{E}k(\tilde{z},\tilde{z}'), \tag{10}$$

estimated with weighted U-statistics on (S_t, S_{t-1}) (Gretton et al., 2012).

2-Wasserstein. If we approximate each coreset by Gaussian (μ_t, Σ_t), the closed form is

$$W_2^2(\mathcal{N}_t, \mathcal{N}_{t-1}) = \|\mu_t - \mu_{t-1}\|_2^2 + \mathrm{Tr}\Big(\Sigma_t + \Sigma_{t-1} - 2\big(\Sigma_{t-1}^{1/2}\Sigma_t\Sigma_{t-1}^{1/2}\big)^{1/2}\Big). \quad (11)$$

Fréchet-type distances. Track Fréchet distance in embedding space (akin to FID) to capture distributional shifts that often precede metric regressions.

5 Practical Implementation and Validation

5.1 Weighted Farthest-First with Difficulty (Geometric Baseline)

Algorithm 1 Weighted Farthest-First with Difficulty Prioritization

Input: data X, features $z(i)$, traffic weights $w(i)$, difficulty $d(i)$, size k, trade-off $\alpha \in [0, 1]$
Output: coreset S and weights v

```
S ← ∅, dist[i] ← +∞
function Priority(i) return α · dist[i] + (1 − α) · d(i)
end function
j_0 ← arg max_i w(i) d(i); add j_0 to S
update dist[i] ← min(dist[i], ‖z(i) − z(j_0)‖) for all i
for t = 2 to k do
    j ← arg max_i Priority(i); add j to S
    update dist[i] ← min(dist[i], ‖z(i) − z(j)‖)
end for
estimate local density / selection probs p_i for i ∈ S
set v_i ← w(i)/p_i                                  ▷ Horvitz–Thompson
```

5.2 Sensitivity (ERM) Coreset via Proxy Gradients

Algorithm 2 Sensitivity / Importance-Sampled ERM Coreset

Input: proxy model f_θ, loss ℓ, features $z(i)$, labels/pseudo-labels y_i, size k
Output: coreset S and weights v

```
fit θ on a cheap subset or using judge labels
compute sensitivity proxy s_i ∝ w_i · ‖∇_θ ℓ(f_θ(x_i), y_i)‖ (or leverage score)
set p_i ∝ s_i with Σ_i p_i = k
sample S by Poisson or VAROPT; set v_i ← w_i/p_i
```

5.3 Streaming Merge-and-Reduce (Scalable)

Algorithm 3 Streaming Merge-and-Reduce

Input: data stream split into shards $X_1, \ldots, X_M$, per-shard size k_1, final size k
Output: coreset S

```
for j = 1 to M do
    build S_j of size k_1 on X_j
end for
S' ← ⋃_j S_j
build final S of size k on S' (reuse Alg. 1 or 2)
```

5.4 Confidence Intervals and Power

With HT weights (5), unbiasedness holds for linear metrics. For non-linear metrics (F1, quantiles), use *delta method* or *paired bootstrap* on S with v_i to form CIs. For A/B deltas Δ_f, use stratified, paired evaluation on the same weighted coreset to reduce variance.

6 Concrete Recipes and Lifecycle Integration

6.1 Recipes

1. **Quick Evaluation Coreset.** Mean-pooled embeddings + tool/outcome/latency/uncertainty; weighted k-center with $k \in [5\text{k}, 20\text{k}]$; top-off with high-uncertainty/cost/policy items; HT weights; paired bootstrap CIs.
2. **Theory-Backed ERM Coreset.** Proxy classifier with pseudo-labels; gradient-norm sensitivities; Poisson/VAROPT sampling; $v_i = w_i/p_i$. Use for retraining judges/reward models or focused failure discovery.
3. **Streaming Drift Watch.** Daily $k \approx 1{,}000$ via merge-and-reduce; alert on MMD^2, W_2^2, or Fréchet distance spiking, and on worst-k slice metrics.
4. **Fairness- or Compliance-Aware.** Stratify by protected/business slices; allocate k_g via Neyman allocation: $k_g \propto N_g \sigma_g$ (estimated within-slice variance), then run any selector in each stratum.

6.2 Lifecycle Integration

- **Pre-Launch:** Assemble a design coreset mixing historical logs, adversarial prompts, and synthetic tool-call chains to span intents/APIs.
- **Canary/A/B:** Compare v_{n+1} vs. v_n on a frozen, versioned coreset for apples-to-apples, plus a recency-weighted coreset for drift sensitivity.
- **HITL Routing:** Send high-v_i and high-uncertainty items to human graders (max expected value of information).
- **Root Cause:** On regression in a slice, re-run local coreset selection within the failing slice to surface minimal counterexamples.

7 Limitations and Open Directions

Embedding myopia. If Z misses crucial causal features (e.g., policy nuances, tool schemas), coverage coresets fail; iterate on Z with explicit tool graphs or retrieved-knowledge features.

Objective mismatch. Coresets preserve what they are asked to preserve. If the business objective g differs from proxy f, bias can appear. Use multi-objective or constrained selection:

$$\min_{S,v} \sum_{r=1}^{R} \lambda_r \left| \hat{f}_X^{(r)} - \hat{f}_S^{(r)} \right| \quad \text{s.t.} \quad \text{slice constraints}, \ |S| = k.$$

Non-linear metrics. For quantiles (e.g., P_{99} latency), use quantile-aware sampling (importance by tail risk) and quantile-specific CI methods.

Adversarial shift. Integrate red-teaming distributions and OOD detection into strata; consider distributionally robust selection via IPMs (e.g., maximize worst-case coverage in Wasserstein balls).

8 Conclusion

Coresets offer a principled bridge between massive interaction logs and practical, trustworthy agent evaluation. With appropriate representations, sampling/selection, and weighting, small weighted subsets enable accurate metrics, stable regression tests, and early drift alerts—at a fraction of cost. The main work is aligning the objective with the business goal, protecting rare slices, and validating rigorously.

References

Agarwal, P. K., Har-Peled, S., & Varadarajan, K. (2012). Merge-and-reduce: A framework for streaming coreset construction. In *SODA*.

Braverman, V., Feldman, D., Lang, H., & Sohler, C. (2016). New frameworks for offline and streaming coreset constructions. *arXiv:1612.00889*.

Campbell, T., & Broderick, T. (2018). Bayesian coreset construction via greedy iterative geodesic ascent. In *ICML*.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., & Sarlós, T. (2012). Fast approximation of matrix coherence and statistical leverage. *JMLR*, 13.

Feldman, D., & Langberg, M. (2011). A unified framework for core-sets. In *STOC*.

Feldman, D., Monemizadeh, M., & Sohler, C. (2013). A PTAS for k-means clustering based on weak coresets. *SODA*.

Gonzalez, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38, 293–306.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., & Smola, A. (2012). A kernel two-sample test. *JMLR*, 13.

Har-Peled, S., & Mazumdar, S. (2018). On coresets for k-means and k-median clustering. *SIAM J. Comp.*, 47(3), 1447–1472.

Houlsby, N., Huszár, F., Ghahramani, Z., & Lengyel, M. (2011). Bayesian active learning for classification and preference learning. *arXiv:1112.5745*.

Huggins, J. H., Campbell, T., & Broderick, T. (2016). Coresets for scalable Bayesian logistic regression. In *NeurIPS* Workshop.

Katharopoulos, A., & Fleuret, F. (2018). Not all samples are created equal: Deep learning with importance sampling. In *ICML*.

Kulesza, A., & Taskar, B. (2012). Determinantal point processes for machine learning. *Foundations and Trends in ML*, 5(2–3).

Li, P., Long, P. M., & Srinivasan, A. (2011). Improved bounds on the sample complexity of learning. *JCSS*, 62(3), 516–526.

Liang, P., Bommasani, R., Lee, T., et al. (2022). Holistic evaluation of language models. *arXiv:2211.09110*.

Lucic, M., Faiss, M., & Krause, A. (2018). Training Gaussian mixture models at scale via coresets. *JMLR*, 18(1).

Mahoney, M. W. (2011). Randomized algorithms for matrices and data. *Foundations and Trends in ML*, 3(2).

Munteanu, A., & Schwiegelshohn, C. (2018). Coresets–methods and history: A theoretician's design pattern for ML. *arXiv:1807.07822*.

Nemhauser, G. L., Wolsey, L. A., & Fisher, M. L. (1978). An analysis of approximations for maximizing submodular set functions—I. *Mathematical Programming*, 14(1), 265–294.