DECISION-FOCUSED ACQUISITION OF PREFERENCE DATA FOR LEARNING INFERENCE-TIME VERIFIERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern Large Language models increasingly use their compute budget for inference-time computation. In applications such as code generation and tool-use, this often involves generating multiple candidate responses and learning a verifier that applies a decision rule such as best-of-K selection, reranking or shortlisting, or escalating to an expert to decide the best response. In such cases, the final utility of the response depends both on the verifier's score and the decision operator applied. In this paper, we consider the problem of active learning to train such verifiers. Existing active learning methods for preference data select the comparisons that most reduce uncertainty in the verifier's parameters, and are indifferent to which directions of parameter space the deployed decision actually consults. Instead, we propose the Decision Focused Acquisition (DFA) criterion, a c-optimal experimental design that targets the decision. We assume access to a small Decision Set where utilities are known and the DFA criterion selects the comparisons that most reduce the posterior variance of a linearised decision loss. This formulation applies to any decision operator that has a differentiable surrogate and we instantiate it for best-of-K selection, top-m shortlisting, and accept-or-escalate cascades. Across four datasets spanning preference reranking, code generation, and tool-use agents, we show that our method outperforms existing active preference learning methods, yielding verifiers with higher decision utility under fixed acquisition budget. Lastly, we show that the benefit of DFA is largest at small and medium budgets, where acquisition is most critical, and that it comes from acquiring comparisons at the decision boundary of the deployed operator.

1 INTRODUCTION

Large language models increasingly spend extra compute at inference time. A generator samples several candidate responses, or reasoning traces, to a prompt, and a verifier selects amongst the candidates under a given decision rule (Cobbe et al., 2021; Lightman et al., 2024; Snell et al., 2024; Brown et al., 2024; Zhang et al., 2025; Liu et al., 2025). Several such decision rules exist at the time of deployment of LLMs. For instance, in mathematics and code, the best-of-K rule returns the candidate that passes the most test cases. Similarly, agentic and tool-use systems generate several tool plans or trajectories and commit to the one that best accomplishes the task (Yao et al., 2024; Pan et al., 2025). In reranking pipelines, the verifier returns a shortlist of best responses and a downstream stage such as a human or a cheaper model picks an output from it (Jiang et al., 2023; Sun et al., 2023; Weller et al., 2025). In cascades and routers, the verifier commits to its answer only when it is confident and otherwise escalates to a human or a stronger LLM (Chen et al., 2023b; Gupta et al., 2024; Yue et al., 2024; Li et al., 2024; Ong et al., 2025; Chuang et al., 2025). However, in practice, such exact verifiers are rarely available when the decision is made. Test cases may be hidden or missing, and whether a task was accomplished may only be known after execution or human review. A verifier is therefore learned to approximate them.

In each of these decision rules, the quality of the system's decision is a function of not just the verifier output but also of the choice of the decision operator. For example, a verifier that inverts many pairs low in the ranking but identifies the top candidate on every prompt is perfect for best-of-K while one that is nearly correct everywhere but misranks the winner is not. Ranking accuracy and decision quality can therefore move in opposite directions. This mismatch requires optimizing the verifier for decision quality rather than intermediate preference rankings. Learning predictors that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

are aligned with the downstream decisions has long been studied in decision-focused learning (Donti et al., 2017; Elmachtoub & Grigas, 2022; Mandi et al., 2022; Wang et al., 2023). However, obtaining true decision utilities, such as by running manually crafted hidden tests or grading a trajectory, is expensive and can only be done for a small curated set of prompts. But a cheap annotator, such as an LLM judge that partly agrees with the true decision, can compare responses at scale. Thus, we have two kinds of data: a small decision set of prompts on which the true decision utility is known, and a large pool of prompts on which preference labels can be acquired from the proxy annotator under a budget. This can be formulated as an active learning problem to decide preference labels to collect so that the verifier trained on them makes good decisions as measured by the true decision utility.

Existing active methods for preference data select comparisons using uncertainty in predicted reward differences (Das et al., 2025; Ji et al., 2024; Lin et al., 2026) or by heuristics such as preference certainty (Muldrew et al., 2024). Another line of work treats preference elicitation as optimal experimental design, and chooses comparisons for accuracy of reward model in every direction of feature space (Mukherjee et al., 2024; Scheid et al., 2024; Thekumparampil et al., 2025). ActiveDPO (Lin et al., 2026), the most recent of these methods, scores pairwise preference data by how uncertain the verifier is about the preference outcome. This is greedy step of classical D-optimal design (Chaloner & Verdinelli, 1995; Atkinson et al., 2007) which shrinks parameter uncertainty in every direction and has no notion of which directions the deployed decision depends on. Thus, the preference labels it spends resolving orderings that the decision operator never consults are wasted.

Instead, we propose an acquisition criterion that chooses comparisons that most reduce the uncertainty about how good the decisions are. This is c-optimal experimental design (Chaloner & Verdinelli, 1995) with $c = g_{\mathrm{dec}}$ (Section 3.2), where g_{dec} is the gradient of a smoothed decision objective. We show in Theorem 1 that this criterion reduces posterior variance of the linearised decision loss under two stated approximations. This construction does not depend on which decision operator is deployed, but requires a differentiable surrogate of the operator. We consider three decision operators commonly used when deploying LLMs, best-of-K selection, top-m shortlisting and accept-or-escalate cascades, together with known differentiable surrogates. Empirically, we show that DFA outperforms existing acquisition baselines in the decision utility achieved on a held-out evaluation set. Its benefit is largest at small budgets, where choosing which labels to acquire matters most. Concretely, our contributions are as follows:

- We formulate data acquisition for learning inference-time verifiers under a decision operator as a c-optimal experimental design problem, where the coefficient vector c is the gradient of a smoothed decision objective on a small decision set with known utilities.
- We derive a Decision-Focused Acquisition (DFA) criterion that reduces the posterior variance of the linearised decision loss, and give practical implementation choices that speed up acquisition by $7.5\times$ on average. To the best of our knowledge, this is the first work to apply decision-focused learning to active preference learning.
- We empirically validate our method on four datasets spanning preference learning, code generation and LLM tool use, and three decision operators, where DFA outperforms existing active preference learning baselines at small and moderate budgets, and retains its advantage even with a very small decision set. We further show that DFA concentrates its labels at the decision boundary of the deployed operator, and that acquiring for a mismatched operator reduces its benefit.

Related work. Our work lies at the intersection of three areas. (i) Inference-time decision rules for LLMs, such as best-of-K selection with learned verifiers, reranking into shortlists and confidence-based cascades (Snell et al., 2024; Zhang et al., 2025; Weller et al., 2025; Chen et al., 2023b; Ong et al., 2025). (ii) active preference learning, which selects comparisons by the uncertainty of the learned preference model or by optimal-design criteria (Muldrew et al., 2024; Das et al., 2025; Lin et al., 2026; Mukherjee et al., 2024; Scheid et al., 2024; Thekumparampil et al., 2025). And (iii) decision-focused learning, which trains predictors against the loss of a downstream decision (Donti et al., 2017; Elmachtoub & Grigas, 2022; Mandi et al., 2022). DFA shifts the decision-focused objective from the training loss to the choice of which preference labels to acquire, for inference-time decision operators in LLMs. Appendix A discusses related work in detail.

2 BACKGROUND

We begin with dataset consisting of prompt $x \in X$ and corresponding K candidate responses $y_1, \ldots, y_K$. Each response also carries a true utility $V^\star(x, y) \in \mathbb{R}$. A verifier $r_\phi(x, y)$ parameterized by ϕ scores a response y for a prompt x. A decision operator $\mathcal{O}$ maps the verifier's scores on the K candidate responses, r_ϕ, to an action $a = \mathcal{O}(x, r_\phi)$ from its action set $\mathcal{A}_\mathcal{O}$, such as selecting a single response, a shortlist of responses, or escalation. The action has a true utility $V^\star(x, a)$, and the regret of the operator on x is

$$\mathrm{Reg}_\mathcal{O}(x; \phi) \;=\; \max_{k \le K} V^\star(x, y_k) \;-\; V^\star(x, a). \tag{1}$$

We partition prompts into three disjoint sets: $\mathcal{T}$, the acquisition pool, where comparisons cost budget and $V^\star$ is never available; $\mathcal{D}$, the decision set, where $V^\star$ is known but which is never queried for comparisons and serves only to define what a good decision is; and $\mathcal{E}$, the evaluation set, used to test performance.

Let A be a set of queries $q = (x, y_i, y_j)$ where $x \in \mathcal{T}$ and y_i, y_j a pair of candidate responses to x. Define the score margin $f_q(\phi) = r_\phi(x, y_i) - r_\phi(x, y_j)$. An acquired label $\ell_q \in \{-1, +1\}$ indicates the annotator's preferred candidate, with $+1$ denoting preference for y_i. We model these labels using Bradley–Terry preferences Bradley & Terry (1952): $P_\phi(\ell_q = +1 \mid q) = \sigma(f_q(\phi))$, where $\sigma(t) = (1 + e^{-t})^{-1}$. Given acquired comparisons A and their labels, we learn the parameters $\hat{\phi}_A$ minimizing the loss

$$L(\phi; A) = \sum_{q \in A} -\log \sigma\big(\ell_q f_q(\phi)\big) + \frac{\lambda}{2}\|\phi\|^2, \qquad \lambda > 0. \tag{2}$$

The objective is to select set of queries A on the acquisition set $\mathcal{T}$, with $|A| \le B$ for a budget B, so as to maximize decision utility on the decision set $\mathcal{D}$

$$A^\star = \arg\max_{|A| \le B} \; J_\mathcal{O}(\phi), \qquad J_\mathcal{O}(\phi) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} V^\star\big(x, a\big). \tag{3}$$

3 DECISION FOCUSED ACQUISITION

Two key challenges exist in obtaining the optimal acquisition set. (i) The decision operator is non-differentiable, so we cannot directly optimise for the decisions. In Section 3.1 we discuss soft, differentiable surrogates of decision operators. (ii) $\hat{\phi}_A$ requires retraining for every candidate A and there are combinatorially large possibilities of A. Instead, in Section 3.2 we provide an acquisition criterion that reduces the uncertainty in decision utility.

3.1 SOFT DECISION OPERATORS

The decision operator in equation 1 outputs a hard action $a = \mathcal{O}(x, r_\phi)$, which is not differentiable in ϕ. Instead, we use soft decision operators $\pi_\phi^\mathcal{O}$, which output a probability $\pi_\phi^\mathcal{O}(a \mid x)$ over actions $a \in \mathcal{A}_\mathcal{O}$. The soft decision utility on the decision set and the corresponding decision loss are

$$J_\tau(\phi) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \sum_{a \in \mathcal{A}_\mathcal{O}} \pi_\phi^\mathcal{O}(a \mid x)\, V^\star(x, a), \qquad L_{\mathrm{dec}}(\phi) = -J_\tau(\phi). \tag{4}$$

We consider three decision operators in our work and provide additional details in Appendix C.

Best-of-K selection. This operator outputs the index of the response with the highest verifier score, $a = \arg\max_k r_\phi(x, y_k)$. We use its standard relaxation, a temperature-τ softmax

$$\pi_k = \frac{\exp\big(r_\phi(x, y_k)/\tau\big)}{\sum_{j=1}^K \exp\big(r_\phi(x, y_j)/\tau\big)}, \tag{5}$$

which recovers the hard operator as $\tau \to 0$.

Top-m shortlist. This operator outputs the indices of the m highest-scoring candidates. We model the utility of this operator as mean of $V^\star$ over the shortlist. Its relaxation uses optimal-transport view of top-k (Xie et al., 2020). Each candidate's unit of mass is transported into two bins, a shortlist of size m and the rest, with higher-scoring candidates preferring the shortlist. Solving this problem with entropic regularisation of strength ε gives the probability that candidate k is in the shortlist,

$$\pi(k \in S \mid x) = \sigma\left(\frac{r_\phi(x, y_k) - t_x}{\varepsilon}\right), \tag{6}$$

As $\varepsilon \to 0$ this recovers the hard top-m shortlist.

Accept or escalate (cascade). This operator outputs the best-of-K candidate by score only if a confidence score s_x is above a threshold t, and otherwise escalates to a human or expert. We relax this into a soft operator over the $K + 1$ actions. The verifier escalates with probability

$$\pi(\text{escalate} \mid x) = 1 - \sigma\left(\frac{s_x - t}{\varepsilon}\right), \tag{7}$$

where we define s_x as the softmax-weighted average score minus the mean score on prompt x, and $\varepsilon \to 0$ gives the hard escalation decision. If not escalated, the operator returns a candidate according to the best-of-K softmax with temperature τ.

3.2 Decision Focused Acquisition Criterion

To measure how uncertain we are about the quality of our decisions, we define a local decision target. Let $\hat{\phi}_A$ be the current fit of the verifier on a given acquisition set A.

Definition 1 (Local decision target)**.** Let $g_{\text{dec}} = \nabla_\phi L_{\text{dec}}(\hat{\phi}_A)$ for the deployed operator's surrogate and define $T = L_{\text{dec}}(\hat{\phi}_A) + g_{\text{dec}}^\top(\phi - \hat{\phi}_A)$, the first-order expansion of the decision loss about the current fit, regarded as a random variable through ϕ.

For a comparison q with score margin f_q, write $v_q = \nabla_\phi f_q(\hat{\phi}_A)$ for the direction in parameter space that the comparison measures, and $w_q = \sigma\big(f_q(\hat{\phi}_A)\big)\big(1 - \sigma\big(f_q(\hat{\phi}_A)\big)\big)$ for the Fisher information that a Bradley–Terry observation of it carries.

The local target T is random only through ϕ, so measuring its uncertainty requires a distribution over the verifier parameters given the comparisons acquired so far.

Assumption 1 (Laplace posterior)**.** We approximate this posterior by a Gaussian centred at the current fit, with covariance given by the inverse Hessian of the training loss,

$$\phi \mid A \sim \mathcal{N}(\hat{\phi}_A, \Sigma), \qquad \Sigma = H_A^{-1}, \qquad H_A = \nabla_\phi^2 L(\hat{\phi}_A; A).$$

Under the Gauss–Newton approximation, the Hessian of the Bradley–Terry loss equation 2 can be written as (Martens, 2020; Bishop, 2006) $H_A = \lambda I + \sum_{q \in A} w_q\, v_q v_q^\top$.

To score a comparison before acquiring it, we also need a model of how its label would update this posterior.

Assumption 2 (Local Gaussian observation model)**.** We model the information carried by comparison q as a linear-Gaussian observation $\tilde{f}_q \mid \phi \sim \mathcal{N}\big(v_q^\top \phi,\ w_q^{-1}\big)$.

Theorem 1 (Decision-Focused Acquisition Criterion)**.** *Under Assumptions 1 and 2, the reduction in the variance of the local decision target from observing comparison q is*

$$\text{Acq}(q) \;=\; \text{Var}\left(T \mid A\right) - \text{Var}\left(T \mid A, \tilde{f}_q\right) \;=\; \frac{w_q \left(g_{\text{dec}}^\top \Sigma v_q\right)^2}{1 + w_q\, v_q^\top \Sigma v_q}. \tag{8}$$

Proof sketch. T is affine in ϕ with fixed coefficients, so $\text{Var}(T \mid A) = g_{\text{dec}}^\top \Sigma g_{\text{dec}}$. Combining the Gaussian posterior of Assumption 1 with the Gaussian likelihood of Assumption 2 adds the observation's precision to the posterior precision, giving the updated covariance $\Sigma_q = (\Sigma^{-1} + w_q v_q v_q^\top)^{-1}$, hence $\text{Var}(T \mid A, \tilde{f}_q) = g_{\text{dec}}^\top \Sigma_q g_{\text{dec}}$. By Sherman–Morrison (Sherman & Morrison, 1950), $\Sigma_q = \Sigma - w_q(\Sigma v_q)(\Sigma v_q)^\top / (1 + w_q v_q^\top \Sigma v_q)$. Substituting and taking the difference in $\text{Var}\left(T \mid A\right)$ and $\text{Var}\left(T \mid A, \tilde{f}_q\right)$ gives acquisition score as $w_q (g_{\text{dec}}^\top \Sigma v_q)^2 / (1 + w_q v_q^\top \Sigma v_q)$. □

Algorithm 1 Decision-Focused Acquisition (DFA)

Input: candidate comparisons Q on $\mathcal{T}$; decision set $\mathcal{D}$ with utilities $V^\star$; decision loss L_{dec} equation 4; budget B; batch size b; number of initial random comparisons n_0
$A \leftarrow n_0$ comparisons drawn uniformly from Q, with their labels
while $|A| < B$ **do**
  $\hat{\phi}_A \leftarrow \arg\min_\phi L(\phi; A)$ ▷ equation 2
  $v_q \leftarrow \nabla_\phi f_q(\hat{\phi}_A),\ w_q \leftarrow \sigma\big(f_q(\hat{\phi}_A)\big)\big(1-\sigma\big(f_q(\hat{\phi}_A)\big)\big)$ for all $q \in Q$ ▷ autodiff
  $\Sigma \leftarrow \big(\lambda I + \sum_{q\in A} w_q v_q v_q^\top\big)^{-1}$ ▷ Assumption 1
  $g_{\mathrm{dec}} \leftarrow \nabla_\phi L_{\mathrm{dec}}(\hat{\phi}_A)$ on $\mathcal{D}$ ▷ closed form, Appendix C
  $\nu_q \leftarrow v_q^\top \Sigma v_q,\ \kappa_q \leftarrow g_{\mathrm{dec}}^\top \Sigma v_q$ for all $q \in Q \setminus A$
  for $j = 1, \ldots, \min(b,\ B - |A|)$ **do**
    $q \leftarrow \arg\max_{q' \in Q\setminus A}\ w_{q'}\kappa_{q'}^2/(1 + w_{q'}\nu_{q'})$ ▷ Theorem 1
    Acquire the label ℓ_q; $A \leftarrow A \cup \{q\}$
    $u \leftarrow \Sigma v_q,\ \beta \leftarrow w_q/(1 + w_q\nu_q),\ \Sigma \leftarrow \Sigma - \beta\, uu^\top$ ▷ equation 9
    Update $\nu_{q'}, \kappa_{q'}$ for all $q' \in Q \setminus A$ ▷ equation 10
  end for
end while
return $\arg\min_\phi L(\phi; A)$

Relation to optimal experimental design. Bayesian experimental design chooses observations to shrink the posterior covariance Σ of the parameters (Chaloner & Verdinelli, 1995). D-optimal design minimises $\det \Sigma$ and thereby reduces uncertainty in every direction of parameter space alike, whereas c-optimal design minimises $c^\top \Sigma c$, the posterior variance of a single linear functional $c^\top \phi$. Since $\mathrm{Var}(T \mid A) = g_{\mathrm{dec}}^\top \Sigma g_{\mathrm{dec}}$, the criterion of Theorem 1 is exactly the greedy step of c-optimal design with $c = g_{\mathrm{dec}}$.

For complete proof see Appendix B.

3.3 Algorithm and Practical Implementation

Algorithm 1 alternates between refitting the verifier and acquiring a batch of b comparisons chosen by equation 8, since refitting after every acquisition would cost one fit per label.

Within a batch, each acquisition must still be reflected in the criterion. Writing $u = \Sigma v_q$ and $\beta = w_q/(1 + w_q\, v_q^\top \Sigma v_q)$ for the comparison just acquired, the Sherman–Morrison identity (Sherman & Morrison, 1950) gives the updated covariance as a rank-one correction at the current fit,

$$\Sigma \ \leftarrow\ \Sigma - \beta\, uu^\top. \tag{9}$$

The criterion equation 8 depends on the covariance only through two statistics of each candidate q': the variance of its margin, $\nu_{q'} = v_{q'}^\top \Sigma v_{q'}$, and its covariance with the decision target, $\kappa_{q'} = g_{\mathrm{dec}}^\top \Sigma v_{q'}$. Recomputing both from the updated covariance costs $O(d^2)$ per candidate, where d is the dimension of ϕ. Instead, we update them for every remaining candidate by substituting equation 9: every remaining q' satisfies

$$\nu_{q'} \ \leftarrow\ \nu_{q'} - \beta\,(v_{q'}^\top u)^2, \qquad \kappa_{q'} \ \leftarrow\ \kappa_{q'} - \beta\,(v_{q'}^\top u)(u^\top g_{\mathrm{dec}}). \tag{10}$$

A pick therefore costs one matrix–vector product for u and one inner product $v_{q'}^\top u$ per candidate comparison. For a pool of $|Q|$ candidate comparisons, this reduces the per-pick cost from $O(|Q|d^2)$ to $O(|Q|d + d^2)$ and leaves the selected comparisons unchanged. Across the four datasets and three decision operators in this work, where the verifier is linear in $d \in \{32, 64\}$ features and the candidate pool holds $|Q|$ between $4{,}200$ and $18{,}674$ comparisons, our implementation of DFA reduces the acquisition time per acquired comparison from 64.5 milliseconds for a naive implementation, which recomputes both statistics from Σ at every pick, to 8.6 milliseconds thus resulting in $7.5\times$ speed-up.

4 EXPERIMENTS

We evaluate DFA on four datasets spanning preference learning, code generation and tool use, under the three decision operators of Section 3.1. For each dataset the K candidates per prompt are fixed, the prompts are split into $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{E}$ as in Section 2, and a weak annotator labels comparisons on $\mathcal{T}$. Section 4.1 describes the datasets, tasks and annotators, and Section 4.2 the baselines, the verifier and the evaluation protocol.

4.1 DATASETS

UltraFeedback. 500 open-ended instructions from UltraFeedback (Cui et al., 2023) with $K = 8$ candidate responses each. True utility is the score of gold reward model for this dataset, OpenAssistant `deberta-v3-large-v2` (Köpf et al., 2023), and we consider the annotator as Qwen3-1.7B.

EvalPlus and BigCodeBench. These are two popular code-generation benchmarks. EvalPlus (Liu et al., 2023) poses self-contained algorithmic functions (375 problems, $K = 16$), and BigCodeBench (Zhuo et al., 2025) poses practical tasks that compose calls to many libraries (577 problems, $K = 12$). The true utility is the fraction of hidden tests a solution passes. The annotator, Qwen3-1.7B, judges which solution looks correct without executing any code.

τ-bench. Multi-turn tool use in the retail domain of τ-bench (Yao et al., 2024). Here, a Qwen3.5-9B agent interacts with a simulated user and a transactional database, with $K = 8$ trajectories rolled out per task from an identical reset (311 tasks). The true utility is the fraction of the reference actions the agent performs, matched on tool and arguments. Because trajectories are long, we consider a stronger annotator Qwen3-4B, which compares two trajectories.

Across all datasets, the annotator preference and the utility are two correlated but distinct signals. Specifically, within-prompt Spearman rank correlation between the annotator scores and $V^\star$, averaged over prompts, is $+0.48$ (UltraFeedback), $+0.28$ (EvalPlus), $+0.17$ (BigCodeBench) and $+0.18$ (τ-bench). This shows that the quantity we are graded on and the quantity we can afford to acquire are similar but not the same measurement, making the acquisition task challenging.

4.2 BASELINES, VERIFIER AND EVALUATION

All baselines share the same acquisition setup and differ only in how comparisons are scored.

Random acquires a uniformly drawn admissible comparison. **Min-margin**, is uncertainty sampling (Settles, 2009; Maystre & Grossglauser, 2017) which scores by $-|f_q(\hat{\phi}_A)|$, and acquires comparisons where the verifier is least certain which candidate is preferred. **Max-margin**, is the preference-certainty rule from Muldrew et al. (2024), which scores by $|f_q(\hat{\phi}_A)|$, and acquires the comparisons on which the verifier is most confident. **D-optimal** is ActiveDPO (Lin et al., 2026) for our problem and scores $v_q^\top \Sigma v_q$. **DFA** is our method from equation 8. We also compare against two budget-independent reference points to calibrate the regret scale: *random candidate* takes the decision with no verifier at all, and *all pairs, oracle labels* fits the verifier on all comparisons in $\mathcal{T}$ labelled by $V^\star$.

We report all results for a verifier that is a linear Bradley–Terry head on frozen features represented by the last-token hidden state of Qwen3.5-9B Qwen Team (2026) for each response. Appendix F repeats the main comparison with frozen features from Gemma-4-E4B Team et al. (2026). Finally, we report the regret equation 1 on $\mathcal{E}$ as mean $\pm$ standard error over 20 seeds. A seed fixes the splits, the pool of comparisons and the annotator labels, so arms sharing a seed differ only in which comparisons they acquire. Further implementation details and hyperparameters are given in Appendix D.

5 RESULTS

Table 1 reports regret after 32 acquired comparisons under all three decision operators from Section 3.1. DFA has the lowest mean in all twelve settings, and the gain over other baselines is statistically significant. To compare the regrets across datasets, we normalise them by the gap between the maximum and minimum regret within a setting. Under best-of-K on UltraFeedback, for instance,

Table 1: Regret on Eval set $\mathcal{E}$ at a budget of $B = 32$ acquired comparisons, under each of the three decision operators of Section 3.1 (lower is better, mean $\pm$ standard error over 20 seeds). Two italic baselines do not depend on the budget and serve as reference points. DFA has the lowest mean regret in all twelve settings. For results on Gemma 4 E4B based verifier, see Appendix F.

Acquisition rule	UltraFeedback	EvalPlus	BigCodeBench	tau-bench
Best-of-K				
Random	1.1723 $\pm$0.0470	0.2008 $\pm$0.0105	0.2465 $\pm$0.0063	0.1737 $\pm$0.0097
Min-margin	1.2688 $\pm$0.0590	0.2222 $\pm$0.0114	0.2581 $\pm$0.0091	0.1798 $\pm$0.0125
Max-margin (Muldrew et al., 2024)	1.2803 $\pm$0.0415	0.2084 $\pm$0.0089	0.2684 $\pm$0.0087	0.1658 $\pm$0.0111
D-optimal (Lin et al., 2026)	1.2068 $\pm$0.0541	0.2158 $\pm$0.0078	0.2481 $\pm$0.0068	0.1692 $\pm$0.0086
DFA (ours)	**0.7699 $\pm$0.0327**	**0.1470 $\pm$0.0059**	**0.2194 $\pm$0.0065**	**0.1049 $\pm$0.0051**
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	0.6190 $\pm$0.0226	0.1113 $\pm$0.0058	0.1880 $\pm$0.0034	0.0837 $\pm$0.0051
Top-3 shortlist, uniform pick				
Random	1.4965 $\pm$0.0265	0.2061 $\pm$0.0075	0.2554 $\pm$0.0056	0.1584 $\pm$0.0065
Min-margin	1.5294 $\pm$0.0343	0.2221 $\pm$0.0100	0.2570 $\pm$0.0064	0.1646 $\pm$0.0077
Max-margin (Muldrew et al., 2024)	1.5820 $\pm$0.0315	0.2065 $\pm$0.0068	0.2675 $\pm$0.0064	0.1530 $\pm$0.0070
D-optimal (Lin et al., 2026)	1.5308 $\pm$0.0366	0.2090 $\pm$0.0067	0.2551 $\pm$0.0046	0.1582 $\pm$0.0055
DFA (ours)	**1.2141 $\pm$0.0176**	**0.1632 $\pm$0.0060**	**0.2363 $\pm$0.0039**	**0.1130 $\pm$0.0042**
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	1.1046 $\pm$0.0157	0.1371 $\pm$0.0050	0.2137 $\pm$0.0026	0.0985 $\pm$0.0031
Accept or escalate, γ=0.7				
Random	1.0611 $\pm$0.0493	0.1755 $\pm$0.0112	0.2230 $\pm$0.0063	0.1273 $\pm$0.0083
Min-margin	1.1446 $\pm$0.0570	0.1977 $\pm$0.0114	0.2222 $\pm$0.0086	0.1328 $\pm$0.0104
Max-margin (Muldrew et al., 2024)	1.1105 $\pm$0.0514	0.1847 $\pm$0.0102	0.2359 $\pm$0.0085	0.1269 $\pm$0.0091
D-optimal (Lin et al., 2026)	1.0678 $\pm$0.0558	0.1789 $\pm$0.0077	0.2268 $\pm$0.0070	0.1252 $\pm$0.0079
DFA (ours)	**0.6690 $\pm$0.0336**	**0.1315 $\pm$0.0063**	**0.2030 $\pm$0.0062**	**0.0854 $\pm$0.0062**
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	0.4605 $\pm$0.0184	0.1004 $\pm$0.0063	0.1593 $\pm$0.0043	0.0750 $\pm$0.0056

this gap is 1.47 wide, and DFA closes 90% of it, while D-optimal only reaches 60%. Across the four datasets DFA closes 74–90% of the gap, and D-optimal acquisition 13–60%.

None of the heuristics performs reliably well by comparison. Min-margin never significantly improves upon random acquisition, and max-margin, the certainty rule of Muldrew et al. (2024), does no better: it is significantly worse than random acquisition in 28 of the 60 dataset–operator–budget settings and never significantly better. We note that min-margin fails because comparisons the verifier is least sure about are often those whose candidates are of equal quality, which carry the least information about the decision. On BigCodeBench at $B = 128$, 59% of the comparisons min-margin acquires are between candidates with identical utility, against 49% for random acquisition and 25% for DFA. We also observe that D-optimal acquisition is not reliably better than random either. On UltraFeedback its regret is higher than or similar to random acquisition's at every budget and operator, which shows the cost of acquiring for an objective that is mismatched with the deployed decision.

How does the regret change with budget and decision set size? To measure how the benefit of DFA over the baselines changes with label budget and with size of decision set, we perform ablations under the best-of-K decision operator and report regret. Figure 1 (top) shows regret at 5 budget values $B \in \{32, 64, 128, 256, 512\}$ under best-of-K for the four datasets. DFA improves significantly over D-optimal in 18 settings, over random in 16, over min-margin in all 20 and over max-margin in 19. All baselines improve with increasing label budget with performance converging to a similar regret at higher budget. Measured as label efficiency, regret DFA reaches with 32 labels is not matched by the best baseline until $B = 256$ on UltraFeedback, BigCodeBench and τ-bench, and is not matched within the sweep on EvalPlus: an $8\times$ saving or more.

To measure the impact of the decision set on the performance of DFA, we test the smallest decision set at which DFA still improves on every baseline. Keeping $\mathcal{T}$, $\mathcal{E}$ and the budget $B = 128$ fixed, we

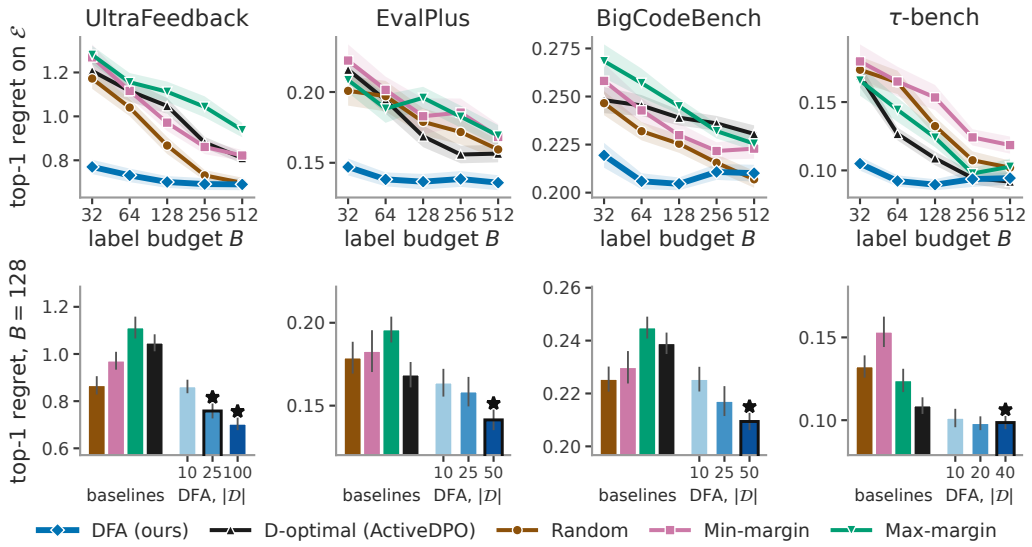

Figure 1: Top-1 regret under best-of-K. Mean over 20 seeds with shaded standard error. *Top:* regret against the label budget B. DFA's advantage is largest where labels are scarcest. *Bottom:* regret at $B = 128$ of each baseline and of DFA with decision sets of increasing size, $\mathcal{T}$ and $\mathcal{E}$ fixed. $\star$ marks sizes at which DFA improves significantly on every baseline. DFA's regret generally decreases as $\mathcal{D}$ grows.

Table 2: Normalized utility under best-of-K decision operator at evaluation (higher is better, mean over 20 seeds). DFA is sensitive to the choice of decision operator as comparisons acquired under a best-of-K operator (matched setting) perform better than comparisons acquired under discard-the-worst, a top-$(K-1)$ shortlist (mismatched setting). Random baseline for comparison.

B	Acquisition	UltraFeedback	EvalPlus	BigCodeBench	τ-bench
128	DFA for best-of-K (matched)	**92.6%**	**64.4%**	**69.1%**	**94.0%**
	DFA for discard-the-worst (mismatched)	85.5%	62.4%	62.8%	86.6%
	Random	79.9%	56.8%	66.2%	45.2%
256	DFA for best-of-K (matched)	**90.8%**	**68.5%**	73.6%	**89.9%**
	DFA for discard-the-worst (mismatched)	87.6%	67.2%	69.7%	82.9%
	Random	86.4%	58.4%	**75.0%**	70.8%

subsample $\mathcal{D}$ to three sizes per dataset, from 10 prompts up to 50% of the full decision set. Figure 1 (bottom) shows the regret at each size next to the baselines, which do not use $\mathcal{D}$. DFA improves significantly on every baseline from $|\mathcal{D}| = 25$ on UltraFeedback, 40 on τ-bench and 50 on EvalPlus and BigCodeBench, and its regret falls as the decision set grows on UltraFeedback and the two code benchmarks. A few prompts with known decision utilities are thus enough to steer acquisition towards the deployed decision, and DFA turns this small signal into a consistent advantage.

How sensitive is DFA to the decision operator? To test how sensitive DFA is to the decision operator, we set up an experiment, in which the verifier is trained on comparisons acquired for the discard-the-worst operator (equivalent to a top-$(K-1)$ shortlist with a uniform pick) but tested under best-of-K. We also report performance when acquisition uses the matched operator, and under random acquisition. Table 2, reports the fraction of the gap between the maximum and minimum regret in different settings. Results show that when DFA acquires for discard-the-worst but is evaluated under best-of-K, its performance drops in every setting, and on BigCodeBench it even drops below that of random acquisition. This shows that DFA acquires the comparisons that are aligned with and specific to the decision operator under which the utilities of $\mathcal{D}$ are evaluated.

EvalPlus
Problem: for each space-separated group of parentheses, return its maximum nesting depth.

DFA acquires

✓ pass 1.00 · rank 1/16

```
for group in paren_string.split():
  stack = []; max_nesting = 0
  for char in group:
    if char == '(':
      stack.append(char)
      max_nesting = max(max_nesting,
                        len(stack))
    elif char == ')':
      stack.pop()
  results.append(max_nesting)
return results
```

× pass 0.01 · rank 9/16

```
def calculate_nesting_level(group):
  ...
    if opening_parentheses == 0:
      return i   # an index, not a depth
  ...
for group in groups:
  result.append(
      calculate_nesting_level(group))
# no return statement
```

D-optimal acquires

× pass 0.01 · rank 9/16

```
def calculate_nesting_level(group):
  ...
    if opening_parentheses == 0:
      return i   # an index, not a depth
  ...
for group in groups:
  result.append(
      calculate_nesting_level(group))
# no return statement
```

× pass 0.01 · rank 9/16

```
for char in paren_string:  # whole input
  if char == '(':
    count += 1
    depth = max(depth, count)
  elif char == ')':
    count -= 1
for char in paren_string:
  if char == ' ':
    result.append(depth)  # same value
  ...                     # for every group
return result
```

Composition of acquired comparisons (EvalPlus, all comparisons acquired up to $B = 128$, 20 seeds)

Best-of-K

Acquired pair	DFA	D-opt.
includes best response	**76%**	39%
both responses in lower half	**6%**	20%

Accept or escalate

Acquired pair	DFA	D-opt.
chosen problem on escalate threshold	**16%**	9%
responses with same pass rate	**7%**	26%

Figure 2: Comparisons acquired by DFA and by D-optimal acquisition on the same EvalPlus problem in the same run, with each candidate's pass rate and rank among the K candidates, and statistics over acquired comparisons under best-of-K and accept-or-escalate. DFA acquires comparison labels for a decision operator near the relevant decision boundary.

What characteristics are shown by acquired comparisons? We log the comparisons acquired by each method together with the true utilities of the two responses, and measure how often (i) a comparison contains the true best response, (ii) both responses lie in the lower half of the true ranking, (iii) the prompt lies on the accept-or-escalate threshold, and (iv) the two responses have equal utility. Figure 2 reports these frequencies for DFA and D-optimal acquisition on EvalPlus, averaged over all comparisons acquired up to $B = 128$ over 20 seeds, together with an example comparison acquired by each method. Under best-of-K, DFA largely compares a correct solution with a failing one. 76% of its comparisons include a best response, against 39% for D-optimal acquisition, which more often compares two responses from the lower half of the ranking (20% against 6%). Under accept-or-escalate, DFA acquires more comparisons on problems whose chosen response lies on the escalation threshold (16% against 9%) and far fewer between responses with the same pass rate (7% against 26%). The two operators place the decision boundary in different places: under best-of-K it separates the best candidate from the rest, whereas under accept-or-escalate it separates the problems that are accepted from those that are escalated. In both cases DFA concentrates its labels near the relevant boundary, which suggests that its gains come from acquiring labels where they can inform the deployed decision. Appendix I reports the same statistics for all four datasets and all three decision operators.

Additional experiments on the effect of the agreement between the annotator and the true utilities, and on a distribution shift between the decision and evaluation sets, are in Appendices H and J.

6 Conclusion

This paper studies how to acquire preference labels for a verifier that is deployed under a known inference-time decision operator, such as best-of-K selection. We propose Decision-Focused Acquisition (DFA), a c-optimal design criterion that reduces the posterior variance of the decision utility. To compute DFA, we consider differentiable surrogates of common decision operators such as best-of-K selection, top-m shortlisting and accept-or-escalate cascades and assume access to a small decision set where decision utilities are known. Across four datasets and three decision operators, DFA outperforms existing acquisition methods at small and moderate budgets, where acquisition matters most, and its advantage grows with the size of the decision set. We also propose practical considerations to improve the computation cost of acquisition scoring. In this work we keep the candidate responses fixed and train only the verifier. Future work may train the generator and the verifier simultaneously and address the challenge of learning from a fixed, offline decision set when the policy generating the candidates keeps changing.

AI USE STATEMENT

In this work, we used generative AI tools to aid in writing the draft and in coding the algorithm and the model training infrastructure.

REPRODUCIBILITY STATEMENT

We have taken several steps to make our results reproducible. The problem setting and all assumptions are stated in Sections 2 and 3.2, and the complete proof of Theorem 1 is given in Appendix B. The decision operators, their differentiable surrogates and the closed-form gradients of the decision loss are described in Section 3.1 and Appendix C. The acquisition procedure is specified in Algorithm 1, and its efficient implementation in Section 3.3. The datasets, candidate generation, true utilities, annotator prompts, verifier features, data splits and hyperparameters are described in Section 4 and Appendix D, and the baselines, the verifier and the evaluation protocol in Section 4.2. Additional results with verifier features from the instruction-tuned Gemma 4 E4B model are given in Appendix F.

REFERENCES

Anthony Atkinson, Alexander Donev, and Randall Tobias. *Optimum experimental designs, with SAS*, volume 34. OUP Oxford, 2007.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical science*, pp. 273–304, 1995.

Jiefeng Chen, Jinsung Yoon, Sayna Ebrahimi, Sercan O Arik, Tomas Pfister, and Somesh Jha. Adaptation with self-evaluation to improve selective prediction in llms. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5190–5213, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023b.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/chuang25b.html.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*, 2023.

Nirjhar Das, Souradip Chakraborty, Aldo Pacchiano, and Sayak Ray Chowdhury. Active preference optimization for sample efficient rlhf. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 96–112. Springer, 2025.

Priya Donti, Brandon Amos, and J Zico Kolter. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems*, 30, 2017.

Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *Management science*, 68(1): 9–26, 2022.

Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *International Conference on Learning Representations*, volume 2024, pp. 4147–4180, 2024.

Kaixuan Ji, Jiafan He, and Quanquan Gu. Reinforcement learning from human feedback with active queries. *arXiv preprint arXiv:2402.09401*, 2024.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. Openassistant conversations-democratizing large language model alignment. *Advances in neural information processing systems*, 36:47669–47681, 2023.

Jia Li, Yuqi Zhu, Yongmin Li, Ge Li, and Zhi Jin. Showing LLM-generated code selectively based on confidence of LLMs. *arXiv preprint arXiv:2410.03234*, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Xiaoqiang Lin, Arun Verma, Zhongxiang Dai, Daniela Rus, See-Kiong Ng, and Bryan Kian Hsiang Low. Activedpo: Active direct preference optimization for sample-efficient alignment. In *International Conference on Learning Representations*, volume 2026, pp. 21618–21635, 2026.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Advances in neural information processing systems*, 36:21558–21572, 2023.

Runze Liu, Junqi Gao, Jian Zhao, Kaiyan Zhang, Xiu Li, Biqing Qi, Wanli Ouyang, and Bowen Zhou. Can 1b llm surpass 405b llm? rethinking compute-optimal test-time scaling. *arXiv preprint arXiv:2502.06703*, 2025.

Jayanta Mandi, Vıctor Bucarey, Maxime Mulamba Ke Tchomba, and Tias Guns. Decision-focused learning: Through the lens of learning to rank. In *International conference on machine learning*, pp. 14935–14947. PMLR, 2022.

James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.

Lucas Maystre and Matthias Grossglauser. Just sort it! a simple and effective approach to active preference learning. In *International Conference on Machine Learning*, pp. 2344–2353. PMLR, 2017.

Subhojyoti Mukherjee, Anusha Lalitha, Kousha Kalantari, Aniket Deshmukh, Ge Liu, Yifei Ma, and Branislav Kveton. Optimal design for human preference elicitation. *Advances in Neural Information Processing Systems*, 37:90132–90159, 2024.

William Muldrew, Peter Hayes, Mingtian Zhang, and David Barber. Active preference learning for large language models. *arXiv preprint arXiv:2402.08114*, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with swe-gym. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi Jaakkola, and Regina Barzilay. Conformal language modeling. In *International Conference on Learning Representations*, volume 2024, pp. 11654–11681, 2024.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Antoine Scheid, Etienne Boursier, Alain Durmus, Michael I Jordan, Pierre Ménard, Eric Moulines, and Michal Valko. Optimal design for reward modeling in rlhf. *arXiv preprint arXiv:2410.17055*, 2024.

Burr Settles. Active learning literature survey. 2009.

Jack Sherman and Winifred J Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21(1):124–127, 1950.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 14918–14937, 2023.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Kiran Koshy Thekumparampil, Gaurush Hiranandani, Kousha Kalantari, Shoham Sabach, and Branislav Kveton. Comparing few to rank many: Active human preference learning using randomized frank-wolfe method. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/thekumparampil25a.html.

Kai Wang, Shresth Verma, Aditya Mate, Sanket Shah, Aparna Taneja, Neha Madhiwalla, Aparna Hegde, and Milind Tambe. Scalable decision-focused learning in restless multi-armed bandits with application to maternal and child health. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 12138–12146, 2023.

Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn J. Lawrie, and Benjamin Van Durme. Rank1: Test-time compute for reranking in information retrieval. *CoRR*, abs/2502.18418, 2025. doi: 10.48550/ARXIV.2502.18418. URL https://doi.org/10.48550/arXiv.2502.18418.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. Differentiable top-k with optimal transport. *Advances in neural information processing systems*, 33:20520–20531, 2020.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *International Conference on Learning Representations*, volume 2024, pp. 21691–21728, 2024.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Seyed Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. In *International Conference on Learning Representations*, volume 2025, pp. 12476–12505, 2025.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pp. 66602–66656, 2025.

A Extended related work

Active preference learning for language models. Muldrew et al. (2024) select prompts by the predictive entropy of the language model and then prefer comparisons on which DPO's implicit preference model is most certain. Das et al. (2025) cast RLHF as a contextual preference bandit and select the contexts and response pairs with the largest uncertainty in the reward difference, bounding the sub-optimality gap of the resulting policy. Ji et al. (2024) query a preference only when this uncertainty exceeds a threshold, with regret and query-complexity guarantees, and apply the idea to DPO. A related line of work casts preference elicitation as optimal experimental design: Mukherjee et al. (2024) extend optimal designs to lists of items with absolute or ranking feedback, Thekumparampil et al. (2025) compute D-optimal designs over K-subset comparisons with a randomised Frank–Wolfe method, and Scheid et al. (2024) select an offline comparison dataset for reward modelling with guarantees on simple regret. ActiveDPO (Lin et al., 2026) selects by an inverse-covariance norm of the difference in gradients of the implicit reward, grounded in neural dueling bandits, to improve the alignment of the language model itself. These methods differ in their heuristics and guarantees, but none conditions acquisition on a decision set with known utilities or adapts it to the decision operator the verifier feeds, and DFA does both.

Decision-focused learning and inference-time decisions in LLMs. A large body of work learns predictors whose outputs feed a downstream optimisation problem and trains them to minimise the loss of the resulting decision rather than the prediction error (decision-focused, or "predict-then-optimise", learning). Applications include stochastic resource scheduling (Donti et al., 2017), combinatorial problems such as shortest paths and portfolio selection (Elmachtoub & Grigas, 2022), which can also be cast as learning to rank (Mandi et al., 2022), and resource allocation in maternal and child health (Wang et al., 2023). Decision operators of this kind are also common in LLMs at inference time. Best-of-K selection with a learned verifier is widely applied at deployment (Cobbe et al., 2021; Lightman et al., 2024; Snell et al., 2024; Zhang et al., 2025), including over agent trajectories (Pan et al., 2025). Top-m shortlisting is used when a ranker narrows the candidates for a second stage, for example when combining the top outputs of several LLMs (Jiang et al., 2023), in LLM re-ranking agents (Sun et al., 2023; Weller et al., 2025), and when LLMs produce set-valued outputs (Quach et al., 2024). Accept-or-escalate operators appear in LLM cascades and routers (Chen et al., 2023b; Gupta et al., 2024; Yue et al., 2024; Ong et al., 2025), in selective prediction with LLMs (Chen et al., 2023a; Chuang et al., 2025), and in verifier-gated acceptance in code assistants (Li et al., 2024). The classical literature learns such predictors by replacing the decision operator with a differentiable surrogate, for instance differentiable sorting and top-k (Grover et al., 2019; Xie et al., 2020), and optimising end to end; the inference-time decision operators of LLMs have received little such study. Moreover, that literature fixes the dataset and changes the training loss, whereas we keep the loss fixed and choose which data to acquire.

B PROOF OF THEOREM 1

Setup. Throughout, A is the current acquired set and $\hat{\phi}_A$ the fit on it. The quantities $g = g_{\mathrm{dec}}$, v_q and w_q are all evaluated at $\hat{\phi}_A$, so they are fixed vectors and scalars, and the only random quantity is ϕ. The proof computes the variance of the local decision target T before and after observing $\tilde{f}_q$, and subtracts the two. The idea is that T is linear in ϕ, so both variances are quadratic forms of g in the posterior covariance, and one Gaussian observation changes that covariance by a rank-one term.

Step 1: variance before the observation. By Definition 1, $T = L_{\mathrm{dec}}(\hat{\phi}_A) + g^\top(\phi - \hat{\phi}_A)$. The first term is a constant, so $T - \mathbb{E}[T \mid A] = g^\top(\phi - \hat{\phi}_A)$, using $\mathbb{E}[\phi \mid A] = \hat{\phi}_A$ from Assumption 1. Hence

$$\mathrm{Var}(T \mid A) = \mathbb{E}\big[g^\top(\phi - \hat{\phi}_A)(\phi - \hat{\phi}_A)^\top g \,\big|\, A\big] = g^\top \,\mathrm{Cov}(\phi \mid A)\, g = g^\top \Sigma g.$$

Step 2: posterior after observing $\tilde{f}_q$. By Bayes' rule, the posterior after the new observation is proportional to the current posterior times the likelihood of $\tilde{f}_q$. The current posterior is $\mathcal{N}(\hat{\phi}_A, \Sigma)$ (Assumption 1) and the likelihood is $\mathcal{N}(v_q^\top \phi, w_q^{-1})$ (Assumption 2), so up to terms that do not depend on ϕ the log posterior is

$$-\tfrac{1}{2}(\phi - \hat{\phi}_A)^\top \Sigma^{-1}(\phi - \hat{\phi}_A) - \tfrac{w_q}{2}\big(\tilde{f}_q - v_q^\top \phi\big)^2.$$

Expanding both squares and collecting terms,

$$-\tfrac{1}{2}\,\phi^\top\big(\Sigma^{-1} + w_q v_q v_q^\top\big)\phi \;+\; \phi^\top\big(\Sigma^{-1}\hat{\phi}_A + w_q \tilde{f}_q\, v_q\big) \;+\; \text{const.}$$

A log density that is quadratic in ϕ is Gaussian, with precision equal to the matrix in the quadratic term. The posterior covariance is therefore

$$\Sigma_q = \big(\Sigma^{-1} + w_q\, v_q v_q^\top\big)^{-1}.$$

The observed value $\tilde{f}_q$ appears only in the linear term, which sets the posterior mean, so Σ_q does not depend on the value of the label.

Step 3: variance after the observation. T is the same affine function of ϕ, with the same fixed g, so the argument of Step 1 applies with the updated covariance:

$$\mathrm{Var}(T \mid A, \tilde{f}_q) = g^\top \Sigma_q g.$$

Step 4: rank-one inverse. Step 2 gives Σ_q as an inverse. To compare it with Σ we use the Sherman–Morrison identity (Sherman & Morrison, 1950): for an invertible matrix M and a vector u,

$$(M + uu^\top)^{-1} = M^{-1} - \frac{M^{-1}uu^\top M^{-1}}{1 + u^\top M^{-1} u}.$$

Take $M = \Sigma^{-1}$ and $u = \sqrt{w_q}\, v_q$, so that $uu^\top = w_q v_q v_q^\top$, $M^{-1} = \Sigma$, $M^{-1}u = \sqrt{w_q}\,\Sigma v_q$ and $u^\top M^{-1} u = w_q\, v_q^\top \Sigma v_q$. Then

$$\Sigma_q = \Sigma - \frac{w_q\,(\Sigma v_q)(\Sigma v_q)^\top}{1 + w_q\, v_q^\top \Sigma v_q}.$$

Step 5: the reduction. Substituting Step 4 into Step 3,

$$g^\top \Sigma_q g = g^\top \Sigma g - \frac{w_q\,(g^\top \Sigma v_q)(v_q^\top \Sigma g)}{1 + w_q\, v_q^\top \Sigma v_q}.$$

Since Σ is symmetric, $v_q^\top \Sigma g = g^\top \Sigma v_q$, which is a scalar. Subtracting from Step 1,

$$\mathrm{Var}(T \mid A) - \mathrm{Var}(T \mid A, \tilde{f}_q) = \frac{w_q\,(g^\top \Sigma v_q)^2}{1 + w_q\, v_q^\top \Sigma v_q},$$

which is equation 8. □

Step 6: an equivalent form. Dividing the numerator and denominator by w_q,

$$\mathrm{Acq}(q) = \frac{\left(g^\top \Sigma v_q\right)^2}{v_q^\top \Sigma v_q + w_q^{-1}} = \frac{\mathrm{Cov}(T, \tilde{f}_q \mid A)^2}{\mathrm{Var}(\tilde{f}_q \mid A)},$$

because $\mathrm{Cov}(T, \tilde{f}_q \mid A) = g^\top \Sigma v_q$ and $\mathrm{Var}(\tilde{f}_q \mid A) = v_q^\top \Sigma v_q + w_q^{-1}$ (the posterior variance of the margin plus the observation noise). This is the usual variance reduction from conditioning one Gaussian variable on another. A comparison is valuable when its outcome is strongly correlated with the decision target, relative to how noisy that outcome is.

Remarks. The reduction is non-negative, since the denominator is at least one. It does not depend on the realised label (Step 2), so a comparison can be scored before it is acquired. Swapping the two responses of a comparison replaces v_q by $-v_q$ and leaves it unchanged.

C Decision operators: relaxations and gradients

For each operator we give the hard rule, its relaxation and the gradient g_{dec} of the decision loss on $\mathcal{D}$. We write $r_k = r_\phi(x, y_k)$ and $V_k = V^\star(x, y_k)$.

Best-of-K. *Operator.* $a = \arg\max_k r_k$. *Relaxation.* The softmax π_k of equation 5 with temperature τ. *Gradient.*

$$g_{\mathrm{dec}} = -\frac{1}{|\mathcal{D}|\,\tau} \sum_x \sum_k \pi_k \big(V_k - \bar{V}_x\big) \nabla_\phi r_k, \qquad \bar{V}_x = \sum_k \pi_k V_k. \tag{11}$$

Top-m shortlist. *Operator.* The m highest-scoring candidates, with utility the mean of $V^\star$ over them. *Relaxation.* The inclusion probabilities $p_k = \sigma\big((r_k - t_x)/\varepsilon\big)$ of equation 6, with the threshold t_x set so that $\sum_k p_k = m$, and $J = \frac{1}{m|\mathcal{D}|} \sum_x \sum_k p_k V_k$. *Gradient.* With $\sigma'_k = p_k(1 - p_k)$, and differentiating t_x implicitly, $\partial t_x / \partial r_j = \sigma'_j / \sum_i \sigma'_i$, so

$$g_{\mathrm{dec}} = -\frac{1}{m\,|\mathcal{D}|\,\varepsilon} \sum_x \sum_k \sigma'_k \big(V_k - \bar{V}_{w,x}\big) \nabla_\phi r_k, \qquad \bar{V}_{w,x} = \frac{\sum_k \sigma'_k V_k}{\sum_k \sigma'_k}. \tag{12}$$

Accept or escalate. *Operator.* Return the best-of-K candidate if the confidence $s_x = \sum_k \pi_k r_k - \frac{1}{K} \sum_k r_k$ exceeds a threshold t, and escalate otherwise; t is set so that a fraction γ of the $N = |\mathcal{D}|$ prompts is accepted. Since the coverage is fixed, the escalation cost is the same for every verifier and the loss is the regret on accepted prompts. *Relaxation.* Prompt x is accepted with probability $a_x = \sigma\big((s_x - t)/\varepsilon\big)$ as in equation 7, with t set so that $\sum_x a_x = \gamma N$, and incurs the smoothed regret $R_x = \max_k V_k - \sum_k \pi_k V_k$. The loss is $L_{\mathrm{dec}} = \frac{1}{N} \sum_x a_x R_x$. *Gradient.* With $\sigma'_x = a_x(1 - a_x)$, and differentiating t implicitly, $\partial t / \partial \phi = \sum_j \sigma'_j \, \partial s_j / \partial \phi \big/ \sum_j \sigma'_j$, so

$$g_{\mathrm{dec}} = \frac{1}{N} \Big[\frac{1}{\varepsilon} \sum_x \sigma'_x \big(R_x - \bar{R}_w\big) \frac{\partial s_x}{\partial \phi} \;+\; \sum_x a_x \frac{\partial R_x}{\partial \phi} \Big], \qquad \bar{R}_w = \frac{\sum_x \sigma'_x R_x}{\sum_x \sigma'_x}, \tag{13}$$

where $\partial R_x / \partial \phi = -\frac{1}{\tau} \sum_k \pi_k (V_k - \bar{V}_x) \nabla_\phi r_k$.

D Experimental details

This appendix gives the details of the four datasets, the annotators, the verifier features, the splits, and the accept-or-escalate operator used in all experiments.

D.1 Candidates and true utilities

UltraFeedback. We use 500 unique instructions from UltraFeedback (Cui et al., 2023). Responses are generated by a mix of Qwen3.5-9B and Llama-3.2-1B-Instruct for diversity, giving $K = 8$ candidates per prompt. The true utility $V^\star$ is the score of the OpenAssistant `deberta-v3-large-v2` reward model (Köpf et al., 2023).

EvalPlus. Candidates are generated by Llama-3.2-3B-Instruct. Eight candidates are sampled at temperature 0.8 and eight at 1.1, giving $K = 16$. The true utility is the fraction of EvalPlus+ test inputs the candidate passes, each run separately under time and memory limits; a candidate that crashes or times out receives $V^\star = 0$.

BigCodeBench. Candidates are generated by Llama-3-8B-Instruct from the instruction prompt of BigCodeBench v0.1.3 (Zhuo et al., 2025), with $K = 12$, temperature 0.8 and top-p 0.95. We keep the tasks whose reference solution passes its own tests. The true utility is the fraction of the task's unit-test methods the candidate passes, with time and memory limits and $V^\star = 0$ on a timeout.

τ-bench. The policy is Qwen3.5-9B with the retail domain's policy document as system prompt and native tool calling, sampled at temperature 0.8 with at most 30 steps. The simulated user is the same model decoded greedily. Each task is rolled out $K = 8$ times from an identical reset of the environment, over 348 retail tasks: all 115 test tasks and 233 of the training tasks. The true utility is an action credit, the mean over the reference actions of the best fraction of arguments matched by an agent call to the same tool, averaged with an output credit, the fraction of required output strings present in the agent's replies, on the tasks that require outputs. A penalty of 0.05 per call to a tool outside the reference, capped at 0.25, is subtracted, and the result is floored at 0.

D.2 Annotator prompts

Pointwise annotator (UltraFeedback, EvalPlus, BigCodeBench). Qwen3-1.7B scores each candidate separately in a single forward pass with thinking disabled. The system prompt is *"You are a strict reviewer. Answer with exactly one word: Yes or No."* and the user message is

```
Task:
{task}

Candidate response:
{candidate}

Question: is this response correct and does it fully satisfy the task?
Answer Yes or No.
```

with the task and candidate each truncated to 6000 characters. The annotator score is the probability of the answer *Yes* as the first token. A comparison between two candidates is labelled by the sign of the difference of their scores. For the code datasets the annotator sees only the problem statement and the code and has no execution environment.

Pairwise annotator (τ-bench). Qwen3-4B compares two trajectories, each given as a compact text digest of its tool calls, observations and replies to the user. The system prompt is *"You are evaluating two attempts at the same task. Answer with exactly one letter: A or B."* and the user message is

```
Task:
{task}

--- Attempt A ---
{trajectory A}

--- Attempt B ---
{trajectory B}

Which attempt better accomplishes the task? Consider whether the required actions were
taken, whether the final state and the response to the user are correct, and whether the
attempt avoided errors and dead ends. Answer A or B.
```

The preference is $P(\mathrm{A})/(P(\mathrm{A}) + P(\mathrm{B}))$ from the first-token probabilities. The order in which the two trajectories are presented is randomised per pair and undone afterwards, so the label does not depend on position.

D.3 Verifier features and dimensionality reduction

For the three text datasets, the feature of a response is the final-layer hidden state of the last token of frozen Qwen3.5-9B, computed on the prompt and response in a fixed task–solution template. For

τ-bench, a trajectory is split into text blocks covering the task with the final reply, the chronological actions and observations, and summary statistics of the trajectory. Each block is encoded by mean-pooling the final-layer states of Qwen3.5-9B and standardised, and numeric statistics of the trajectory, such as its length and number of tool calls, are appended. In every dataset the features are then projected by principal component analysis, fit on the candidates of the acquisition set $\mathcal{T}$ only, onto the leading d components, each scaled to unit variance. We use $d = 64$ for UltraFeedback and EvalPlus and $d = 32$ for BigCodeBench and τ-bench. The verifier r_ϕ is linear in these d features. Nothing in DFA is specific to linear reward functions: the criterion equation 8 needs only the gradients v_q and g_{dec} and the Gauss–Newton posterior over the trained parameters. Due to compute limitations we run all experiments with a linear verifier, but the method extends to training large reward models, for example with LoRA fine-tuning, where ϕ are the LoRA parameters.

D.4 Splits and comparison pool

We drop the prompts on which $V^\star$ or the annotator is constant across all K candidates, since no comparison on them carries information. Table 3 gives the remaining sizes. For every seed the remaining prompts are permuted and split into $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{E}$ of the sizes in the table. The admissible comparisons are all pairs of candidates within a prompt of $\mathcal{T}$, except pairs of byte-identical candidates, which have identical features. Every arm starts from the same 16 comparisons drawn uniformly at random and then acquires batches of $\min(8, B - |A|)$ comparisons, so the budget is met exactly. Algorithm 1 is thus run with $n_0 = 16$ and $b = 8$, and the surrogates use $\tau = 0.3$ and $\varepsilon = \tau$. Budgets range over $B \in \{32, 64, 128, 256, 512\}$. We call a paired difference between two acquisition rules significant when it exceeds twice its paired standard error over seeds.

Table 3: Dataset sizes after dropping prompts on which $V^\star$ or the annotator is constant, and the per-seed splits. d is the number of principal components the verifier is fit on.

Dataset	Prompts	K	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{E}\rvert$	Comparisons in $\mathcal{T}$	d	Annotator
UltraFeedback	500	8	200	200	100	5,600	64	Qwen3-1.7B, pointwise
EvalPlus	375	16	170	110	95	20,400	64	Qwen3-1.7B, pointwise
BigCodeBench	577	12	200	130	247	13,200	32	Qwen3-1.7B, pointwise
τ-bench	311	8	150	90	60	4,200	32	Qwen3-4B, pairwise

The comparison counts are before byte-identical pairs are removed. On τ-bench $|\mathcal{E}|$ is capped at 60 prompts.

D.5 Accept-or-escalate operator

Escalation cost and coverage. Per-prompt regret is the best-of-K regret if the verifier accepts and an escalation price c if it escalates. The price never has to be chosen: we fix the coverage γ, the fraction of prompts accepted, which pins $\sum_x (1 - a_x)$, so the escalation term is the same constant for every verifier and drops out. What remains is the selective regret on the accepted prompts, which is what we report, and $\gamma = 1$ recovers best-of-K exactly. We use $\gamma = 0.7$.

Threshold. After each fit, the threshold t is the $(1 - \gamma)$ quantile of the confidence statistic over the prompts of $\mathcal{D}$, and it is applied unchanged to $\mathcal{E}$, so no quantity is fit on $\mathcal{E}$.

Confidence statistic. For acquisition, DFA uses the smooth statistic of Section 3.1, $s_x = \sum_k \pi_k r_\phi(x, y_k) - \frac{1}{K} \sum_k r_\phi(x, y_k)$, with $\varepsilon = \tau = 0.3$.

E Hardware

All experiments were run on a SLURM cluster. Generating the candidate responses, scoring them with the annotator, and computing the frozen verifier features of the responses were run on NVIDIA A100 GPUs, with one GPU per job. The acquisition experiments use only these precomputed features and annotator scores, and were run on CPUs, parallelised over datasets, acquisition rules, budgets and seeds.

F MAIN RESULTS WITH GEMMA 4 E4B

Table 4 repeats the comparison of Table 1 with the verifier's frozen features taken from Gemma 4 E4B (Team et al., 2026) instead of Qwen3.5-9B. Candidates, utilities, annotator labels and splits are unchanged. Since only $B = 32$ is run, λ is selected on the $B = 32$ mean. DFA has the lowest regret among random, min-margin, D-optimal and DFA in all twelve settings.

Table 4: Replication of Table 1 with a different frozen feature extractor: Gemma-4-E4B instead of Qwen3.5-9B. Regret on $\mathcal{E}$ at $B = 32$ under each decision operator; mean $\pm$ standard error over 20 seeds. Candidates, utilities, annotator labels and splits are identical to Table 1; only the verifier's features change. †: DFA is better than every other acquisition rule in the table by more than twice the paired standard error.

Acquisition rule	UltraFeedback	EvalPlus	BigCodeBench	τ-bench
Best-of-K				
Random	1.5246 $\pm$0.0397	0.2408 $\pm$0.0097	0.2878 $\pm$0.0074	0.1675 $\pm$0.0117
Min-margin	1.5103 $\pm$0.0532	0.2386 $\pm$0.0096	0.2880 $\pm$0.0058	0.1733 $\pm$0.0089
D-optimal (Lin et al., 2026)	1.4986 $\pm$0.0524	0.2361 $\pm$0.0097	0.2843 $\pm$0.0049	0.1532 $\pm$0.0091
DFA (ours)	**0.9181** $\pm$**0.0288**†	**0.1717** $\pm$**0.0069**†	**0.2426** $\pm$**0.0028**†	**0.1104** $\pm$**0.0069**†
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	0.7033 $\pm$0.0235	0.1629 $\pm$0.0062	0.2384 $\pm$0.0032	0.0863 $\pm$0.0044
Top-3 shortlist, uniform pick				
Random	1.7112 $\pm$0.0250	0.2404 $\pm$0.0074	0.2889 $\pm$0.0038	0.1595 $\pm$0.0074
Min-margin	1.6908 $\pm$0.0318	0.2384 $\pm$0.0070	0.2892 $\pm$0.0037	0.1617 $\pm$0.0061
D-optimal (Lin et al., 2026)	1.7135 $\pm$0.0271	0.2323 $\pm$0.0056	0.2866 $\pm$0.0034	0.1515 $\pm$0.0065
DFA (ours)	**1.3202** $\pm$**0.0263**†	**0.1924** $\pm$**0.0061**†	**0.2569** $\pm$**0.0020**†	**0.1153** $\pm$**0.0044**†
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	1.1705 $\pm$0.0167	0.1738 $\pm$0.0052	0.2504 $\pm$0.0026	0.0981 $\pm$0.0027
Accept or escalate, γ=0.7				
Random	1.3811 $\pm$0.0428	0.2274 $\pm$0.0135	0.2724 $\pm$0.0075	0.1264 $\pm$0.0101
Min-margin	1.3965 $\pm$0.0636	0.2251 $\pm$0.0105	0.2711 $\pm$0.0055	0.1318 $\pm$0.0112
D-optimal (Lin et al., 2026)	1.3038 $\pm$0.0648	0.2114 $\pm$0.0093	0.2698 $\pm$0.0058	0.1263 $\pm$0.0097
DFA (ours)	**0.7256** $\pm$**0.0367**†	**0.1560** $\pm$**0.0084**†	**0.2275** $\pm$**0.0038**†	**0.0790** $\pm$**0.0059**†
Random candidate (no verifier)	2.0878 $\pm$0.0178	0.2797 $\pm$0.0041	0.3107 $\pm$0.0021	0.1818 $\pm$0.0043
All pairs, oracle labels	0.5382 $\pm$0.0208	0.1320 $\pm$0.0067	0.2246 $\pm$0.0044	0.0726 $\pm$0.0042

G DECISION-SET SIZE AT BOTH BUDGETS

Figure 3 repeats the decision-set sweep of Figure 1 (bottom) at $B = 32$ (top row) alongside $B = 128$ (bottom row). As in the main text, $\mathcal{T}$ and $\mathcal{E}$ are the splits of Table 1, only $\mathcal{D}$ is subsampled. Table 5 gives the smallest decision set at which DFA improves on each baseline by more than twice the paired standard error. Against D-optimal acquisition DFA wins 18 of the 20 cells at $B = 32$ and 15 of 20 at $B = 128$, and loses none. Against random acquisition it wins 16 and 14 and also loses none; on the two code datasets at $B = 128$ random acquisition is a strong baseline and DFA needs $|\mathcal{D}| = 50$ to beat it.

H EFFECT OF THE CORRELATION BETWEEN ANNOTATOR AND TRUE UTILITIES ON DFA

DFA learns the decision-relevant direction from the decision set, but the only information it receives about the acquisition prompts comes from the annotator's labels. We therefore degrade the annotator on UltraFeedback, whose real annotator has the highest correlation with $V^\star$ among our datasets (0.48), and ask how weak it can become before DFA stops helping. At zero correlation the labels carry no information about $V^\star$, so no acquisition rule should beat random acquisition.

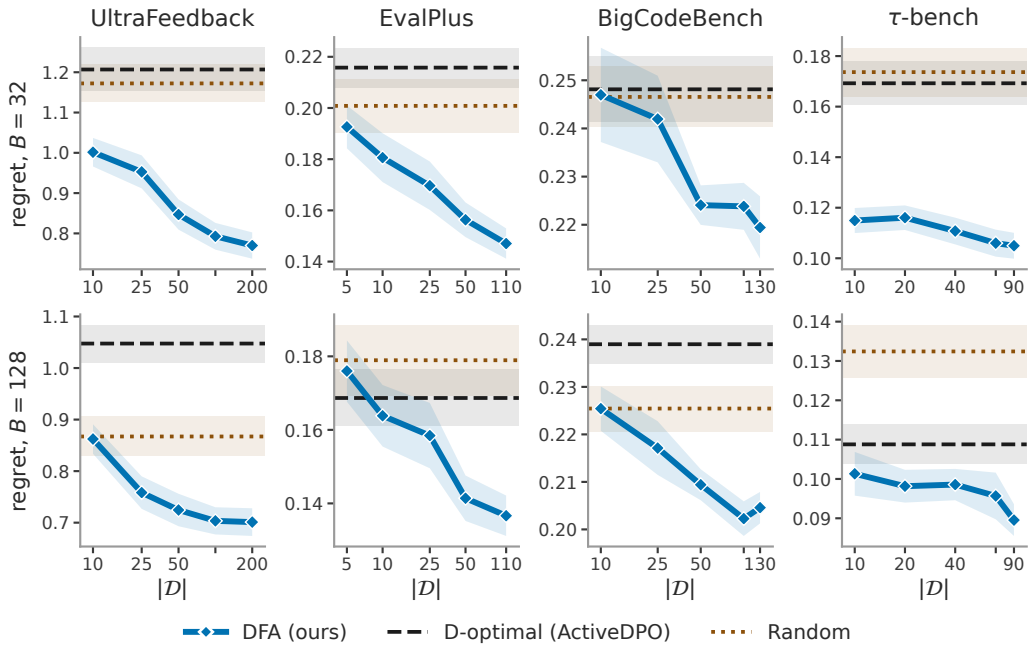

Figure 3: Top-1 regret against the size of the decision set at $B = 32$ (top) and $B = 128$ (bottom), with $\mathcal{T}$ and $\mathcal{E}$ fixed; mean over 20 seeds with shaded standard error. The bottom row extends Figure 1 (bottom) to the full decision set of Table 1.

Table 5: Smallest decision set $|\mathcal{D}|$ at which DFA improves on each baseline by more than twice the paired standard error (20 seeds). Sizes tested: $\{10, 25, 50, 100, 200\}$ on UltraFeedback, $\{5, 10, 25, 50, 110\}$ on EvalPlus, $\{10, 25, 50, 100, 130\}$ on BigCodeBench and $\{10, 20, 40, 70, 90\}$ on τ-bench; the largest is the decision set of Table 1. Once DFA is significantly better, it stays so at every larger size.

Budget	Baseline	UltraFeedback	EvalPlus	BigCodeBench	τ-bench
$B = 32$	D-optimal	10	5	50	10
	Random	10	25	50	10
$B = 128$	D-optimal	10	50	10	40
	Random	25	50	50	10

Degrading the annotator. The annotator scores each candidate and a comparison is labelled by the sign of the score difference. We replace the score by $\sqrt{1-w}\,z+\sqrt{w}\,\xi$, where z is the candidate's within-prompt rank under the annotator, standardised to zero mean and unit variance, and $\xi \sim \mathcal{N}(0,1)$ is drawn once per seed, prompt and candidate and shared by every acquisition rule, so the comparison between rules stays paired. At $w = 0$ this is the original annotator and at $w = 1$ the labels are independent of $V^\star$. We choose w so that the within-prompt Spearman correlation between the degraded scores and $V^\star$, the statistic of Section 4.1, takes the values 0.48 (the original annotator), 0.40, 0.30, 0.20, 0.10, 0.05 and 0. Everything else is as in Table 1 under best-of-K, with 50 seeds instead of 20 to tighten the zero-correlation comparison.

Results. Figure 4 shows the result. DFA keeps its advantage over a wide range of annotator quality. Its regret reduction over D-optimal acquisition at $B = 32$ is 0.46 with the original annotator and stays between 0.40 and 0.53 down to a correlation of 0.20, while all three baselines degrade in parallel. DFA is still significantly better than both random and D-optimal acquisition at a correlation of 0.10, a fifth of the original, at both budgets (by 0.20 and 0.21 at $B = 32$). At 0.05 the difference

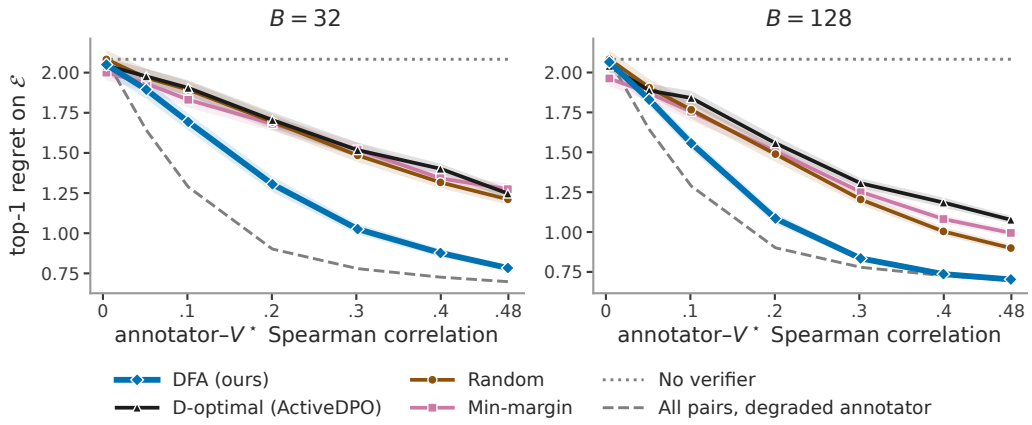

Figure 4: Regret on UltraFeedback as the annotator is degraded, at $B = 32$ (left) and $B = 128$ (right); mean over 50 seeds with shaded standard error. The horizontal axis is the within-prompt Spearman correlation between the annotator and $V^\star$; 0.48 is the original annotator. Dotted: no verifier. Dashed: the verifier fit on every comparison in $\mathcal{T}$ labelled by the degraded annotator ($\phi^\star$ at that noise level).

still favours DFA but is within noise. A weaker annotator used well can beat a stronger one used indiscriminately: at $B = 128$, DFA with an annotator of correlation 0.30 reaches a regret of 0.83, lower than random acquisition with the original annotator (0.90). DFA also stays close to $\phi^\star$, the best this verifier can do with a given annotator: at $B = 128$ the two are within 0.06 for every correlation of 0.30 or more.

At zero correlation every rule, DFA included, is within 0.13 of the regret of taking a decision with no verifier (2.08). DFA differs from random acquisition by -0.03 ± 0.07 at $B = 32$ and -0.02 ± 0.07 at $B = 128$, and from D-optimal acquisition by $+0.01 \pm 0.06$ and $+0.02 \pm 0.06$, none of them significant. The decision set therefore improves DFA only by directing which comparisons are acquired; it gives no advantage when the acquired labels are uninformative.

I Characteristics of acquired annotator labels

Section 5 characterises the comparisons DFA acquires. Table 6 reports the same statistics for every dataset and decision operator. We replay each acquisition run of Table 1 (20 seeds) up to $B = 128$ and log every comparison acquired after the 16 random seed comparisons. For each acquired comparison we record whether the annotator's label agrees with the true utility, whether one of the two candidates is a best candidate of its prompt, whether the two candidates have equal utility, and whether both lie in the bottom half of their prompt's ranking.

The pattern of Section 5 holds in every setting. The annotator agrees with $V^\star$ on 77–91% of DFA's acquired comparisons, against 67–82% for D-optimal and 57–69% for random acquisition, and the difference to D-optimal is significant in 10 of the 12 settings. DFA acquires fewer comparisons between tied candidates in every code and tool-use setting (six to nine percent against 22–26% for D-optimal on EvalPlus and τ-bench), fewer between two poor candidates everywhere except τ-bench, where the two are similar, and in all 12 settings more of its acquired comparisons include a best candidate than D-optimal's, significantly so in 9. Under the top-3 shortlist, the operator-specific statistic is whether a comparison straddles the true top-3 boundary, one candidate inside it and one outside, which is exactly where a shortlist decision is made: 85, 79, 59 and 67% of DFA's acquired comparisons do so on UltraFeedback, EvalPlus, BigCodeBench and τ-bench, against 69, 46, 54 and 52% for D-optimal and at most 53% for random acquisition, significant in all four datasets.

Table 6: Acquisition statistics, in percent of comparisons acquired after the 16 random seed comparisons, up to $B = 128$; mean over 20 seeds. **Bold**: DFA is better than D-optimal acquisition by more than twice the paired standard error. Random and D-optimal acquisition do not consult the operator, so their columns repeat across blocks. UltraFeedback utilities are continuous reward-model scores, so ties do not occur there.

		Annotator agrees with $V^\star \uparrow$			Includes a best candidate $\uparrow$			Tied utilities $\downarrow$			Both in bottom half $\downarrow$		
Operator	Dataset	Rand.	D-opt.	DFA	Rand.	D-opt.	DFA	Rand.	D-opt.	DFA	Rand.	D-opt.	DFA
Best-of-K	UltraFeedback	69	82	**90**	24	38	**43**	0	0	0	21	9	**5**
	EvalPlus	64	74	**87**	58	39	**76**	41	26	**7**	10	20	**6**
	BigCodeBench	59	67	**77**	56	51	53	49	29	25	7	10	**7**
	τ-bench	57	76	77	59	40	**49**	37	22	**9**	8	15	14
Top-3 shortlist	UltraFeedback	69	82	**91**	24	38	**43**	0	0	0	21	9	**4**
	EvalPlus	64	74	**88**	58	39	**78**	41	26	**6**	10	20	**6**
	BigCodeBench	59	67	**80**	56	51	53	49	29	**25**	7	10	**7**
	τ-bench	57	76	**84**	59	40	**49**	37	22	**8**	8	15	15
Accept or escalate	UltraFeedback	69	82	**89**	24	38	**43**	0	0	0	21	9	**5**
	EvalPlus	64	74	**86**	58	39	**76**	41	26	**7**	10	20	**6**
	BigCodeBench	59	67	**80**	56	51	52	49	29	25	7	10	**8**
	τ-bench	57	76	77	59	40	**49**	37	22	**9**	8	15	15

Table 7: Distribution shift on EvalPlus: $\mathcal{T}, \mathcal{D}$ from MBPP+ and $\mathcal{E}$ from HumanEval+, against a matched same-distribution control with $|\mathcal{E}|$ equal. Top-1 regret, 20 seeds. Shift raises the regret of every arm; DFA's margin over D-optimal survives it.

$\lvert\mathcal{D}\rvert$	Condition	Random	Min-margin	D-optimal	DFA
25	matched	0.1990	0.2138	0.2215	**0.1689**
	shifted	0.2236	0.2517	0.2368	**0.1823**
50	matched	0.2001	0.2130	0.2238	**0.1601**
	shifted	0.2236	0.2517	0.2368	**0.1788**

J Robustness to distribution shift between the decision and evaluation sets

Every split so far draws $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{E}$ from one pool, so the decision set is a faithful sample of what the system will face at evaluation. We consider the real world scenario where the evaluation may drift from decision set. EvalPlus is naturally suited for this setting. Its problems come from two sources, MBPP+ and HumanEval+, which differ in provenance, length and style. We draw $\mathcal{T}$ and $\mathcal{D}$ from MBPP+, with $|\mathcal{T}| = 100$, and evaluate on HumanEval+, against a matched control that draws all three from the pooled set with $|\mathcal{E}|$ held equal.

Table 7 reports $B = 32$. We note that the shift hurts all methods. Regret rises for every arm, by up to 0.039, making evaluation harder. DFA however remains the best arm in both conditions and by a similar margin of 0.053 over D-optimal matched and 0.055 shifted at $|\mathcal{D}| = 25$, 0.064 and 0.058 at $|\mathcal{D}| = 50$. The advantage from DFA therefore persists under distribution shift.