DISJUDGE: BUYING RELIABLE ADJUDICATION WHERE CANDIDATES DISAGREE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models often write several candidate programs for one task, and a common way to select one of them is to count the generated tests that each candidate passes. The expected outputs of these tests come from an oracle, which is usually a model and often the one that wrote the candidates. We show that the count compares two candidates only on the inputs where they disagree, so only the accuracy of the oracle on this disagreement set matters, and its overall accuracy can hide a strong bias there. On six code benchmarks, a same-model oracle is right on 87.3% of all inputs, yet on the disagreement set it returns the output of the correct candidate 37.2% of the time and that of a wrong candidate 51.1% of the time. We prove that for a pair of candidates the count selects the wrong one with probability tending to one whenever the oracle favors it on this set, whereas a calibrated likelihood score stays consistent at a rate set by the Hellinger distance between the answers of the oracle under the two hypotheses. Based on this analysis we propose DisJudge, which spends a small budget of reliable answers on the disagreement set and ranks the candidates by the calibrated likelihood of all answers. With 0.2 reliable calls per disagreement input, DisJudge raises the mean selection accuracy over 18 benchmark and generator settings from 54.8% to 60.5%, while the best of 21 baseline rules reaches 57.6%. The bias, and with it the gain of DisJudge, shrinks as the oracle shares less with the generator.

1 INTRODUCTION

Large language models can write many candidate programs for one task, and the chance that at least one of them is correct grows quickly with the number of samples (Chen et al., 2021; Li et al., 2022; Brown et al., 2024). Turning this into a single answer requires a selection rule. A widely used rule, the count, runs every candidate on a set of test inputs and keeps the one that passes the most tests (Li et al., 2022; Chen et al., 2023). Each test is an input together with its expected output. When no reference solution exists, which is the usual case in deployment, the expected outputs are also written by a model, and often by the same model that wrote the candidates (Chen et al., 2023; Huang et al., 2023; Schäfer et al., 2024). Following the software testing literature, we call the source of expected outputs the oracle (Barr et al., 2015). The count relies on these expected outputs, and we ask which property of the oracle decides whether the count selects a correct program.

Consider first how the count compares any two candidates. On an input where both candidates return the same output, the oracle credits both or neither, so the input cancels from the difference of their counts. Only the inputs where the two candidates disagree can change their order; we call them the disagreement set D, after the disagreement region of active learning (Cohn et al., 1994; Hanneke, 2014). Let T be the number of tests, $\Pr(D)$ the probability that a test input falls in D, and $p_\star$ and p_{wrong} the probabilities that the oracle returns the output of the correct and the wrong candidate on D. The expected margin of the correct candidate over the wrong one is then $T \Pr(D)(p_\star - p_{\text{wrong}})$ (Section 3). The accuracy of the oracle outside D does not appear in this margin. The usual overall accuracy mixes the two parts and, because D is small, is dominated by the part that the ranking never uses, so an oracle can be accurate overall and still favor the wrong program on D.

This happens in practice. On six code benchmarks, an oracle written by the generator itself is right on 87.3% of all test inputs, yet on D it returns the correct output only 37.2% of the time and the output of the wrong candidate 51.1% of the time (Fig. 1, Table 1). The bias $\mu = p_{\text{wrong}} - p_\star$ is

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.

positive for 31.0% of candidate pairs, and about a third of all wrong selections are attributed to bias on D. From 10 to 200 tests, the accuracy of the count rises by less than half a point. The bias depends on how much the oracle shares with the generator. From the same model and prompt to a model of another family, μ falls from $+0.139$ to -0.063, and the rate at which the oracle reproduces the exact output of a wrong candidate falls from 18.6% to 2.4%, while the overall accuracy remains between 87% and 89%. The oracle shares errors with the candidates, and its overall accuracy does not reveal them. An answer that corrects the bias must come from a source that does not share these errors, such as an executable reference or a human, and it is usually scarce.

With few reliable answers, a method must choose where to ask and how to weigh each one. By the cancellation above, a reliable answer outside D changes nothing, so every call belongs in D. Which input of D to ask about has a classical answer: the input that splits the remaining candidates most evenly, as in generalized binary search (GBS) (Dasgupta, 2004; Nowak, 2011) and oracle-guided synthesis (Jha et al., 2010). How to use the answers is less settled. Adding each answer as one more test, the vote, gives every answer the same weight, and hard elimination lets one answer remove a candidate permanently. Neither rule asks how reliable an answer is, and both keep the answers of the oracle at face value. A biased oracle, however, still carries information: once its confusion on D is calibrated, its agreement with a candidate is evidence about that candidate, and when $\mu > 0$ it is evidence against it. We prove that for a pair of candidates the count selects the wrong one with probability tending to one whenever $\mu > 0$, while a calibrated likelihood score is consistent whenever $p_\star \neq p_{\mathrm{wrong}}$, at a rate set by the Hellinger distance between the answers of the oracle under the two hypotheses (Theorem 1). The vote is the special case of a source with one constant error rate, and hard elimination is the limit of a vanishing one.

This analysis leads to DisJudge, which spends reliable calls only on D, orders them by GBS, and ranks the candidates by the calibrated likelihood of all answers on D, from the oracle and from the reliable source. We compare it with 21 baseline rules on six benchmarks and candidate pools from three generators, at a matched budget of 0.2 reliable calls per disagreement input. DisJudge raises the mean selection accuracy from 54.8% for the count to 60.5%, the best baseline rule reaches 57.6%, and the paired gain over the count is positive in all 18 settings. A controlled simulation shows that each part of the method adds accuracy, and that hard elimination keeps a correct candidate at close to the rate $(1-\varepsilon)^m$ of m independent calls with error ε. Among the four model-written oracles, DisJudge is most accurate under the most biased one and least accurate where the Hellinger distance is smallest. Our contributions are an analysis that attributes a third of the wrong selections to the bias of the oracle on the disagreement set, a consistency result for the count and for calibrated scores, and a selection rule that uses a small budget of reliable answers.

2 Related Work

Selecting among generated programs. Execution-based rules compare candidates through their outputs on generated inputs: CodeT scores candidates by their agreement with generated tests (Chen et al., 2023), AlphaCode clusters candidates by behavior (Li et al., 2022), MBR-Exec minimizes an execution-based risk (Shi et al., 2022), and self-consistency keeps the most common output (Wang et al., 2023). Learned verifiers and rankers score candidates with trained models (Ni et al., 2023; Zhang et al., 2023; Inala et al., 2022; Cobbe et al., 2021; Lightman et al., 2024), language models act as judges or critics (Zheng et al., 2023; Saunders et al., 2022), and agentic and test-time scaling methods generate tests, debug and select in turn (Huang et al., 2023; Chen et al., 2024; Li et al., 2025). All of them rely on outputs or scores that a model produces and differ in how they aggregate them. We evaluate 15 of these rules, together with three query rules from active learning, on shared candidate pools, and we place DisJudge after four of them. Our question is complementary: which property of the expected outputs decides these rules, and where a few reliable answers should go.

Models that share the errors of the generator. Language-model evaluators prefer their own generations (Panickssery et al., 2024; Wataoka et al., 2024; Zheng et al., 2023), models from the same family make correlated errors (Kim et al., 2025; Goel et al., 2025), and a model often fails to correct its own reasoning (Huang et al., 2024). These studies measure agreement over whole outputs or whole tasks. We locate the effect on the inputs where candidates disagree, show that overall accuracy can hide it, and show that it becomes usable evidence once calibrated.

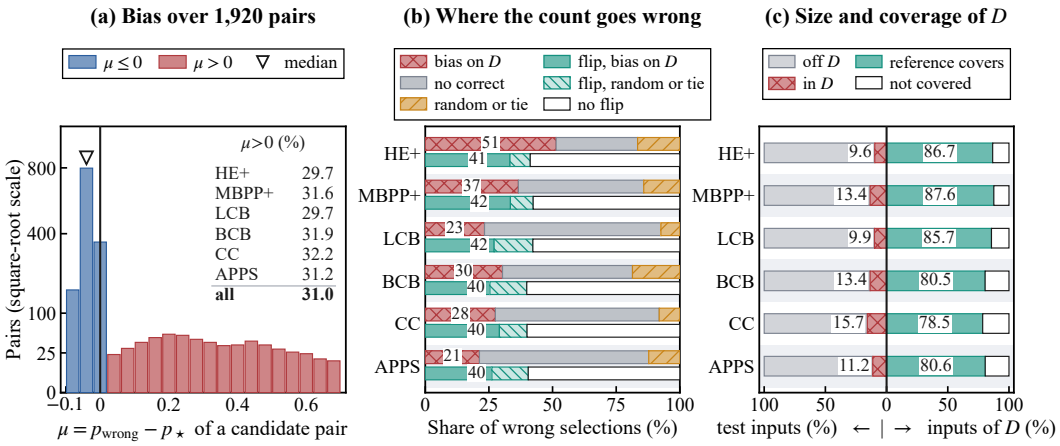

Figure 1: Bias of a same-model oracle on the disagreement set D, six benchmarks (HE+: HumanEval+, LCB: LiveCodeBench, BCB: BigCodeBench, CC: CodeContests), Qwen2.5-Coder-32B pools with $n = 20$ candidates and $T = 100$ tests. (a) Distribution of $\mu = p_{\text{wrong}} - p_\star$ over the 1,920 candidate pairs, 320 per benchmark (square-root scale; ∇ marks the median); the list gives the share of pairs with $\mu > 0$ per benchmark, 31.0% in total. (b) Upper bar of each benchmark: all wrong selections of the count, split into bias on D ($\mu > 0$ for the selected and a correct candidate), pools without a correct candidate, and random errors or ties. Lower bar: wrong selections in pools that hold a correct candidate; the green part becomes correct when the answers of the oracle on D are replaced by the reference, split by the same rule into bias and random flips. (c) Share of the test inputs that lie in D (left) and share of D that the executable reference covers (right).

Aggregating noisy answers and choosing queries. Estimating the confusion matrix of each annotator and weighting its answers accordingly goes back to Dawid & Skene (1979) and learning from crowds (Raykar et al., 2010), and the log-odds weighted majority is optimal for independent binary voters (Nitzan & Paroush, 1982; Grofman et al., 1983). Choosing which input to ask about is the subject of active learning (Settles, 2009), including query by committee (Seung et al., 1992), generalized binary search and its noisy variant (Dasgupta, 2004; Nowak, 2009; 2011), and Bayesian active learning with noisy observations (Golovin et al., 2010). In program synthesis, distinguishing inputs separate candidate programs (Jha et al., 2010; Ji et al., 2020), and interactive code generation asks the user about tests (Lahiri et al., 2022; Fakhoury et al., 2024). The query rule and the log-odds weighting come from this line of work. Our contribution on this side is the analysis of how the answers of a biased oracle enter a calibrated score and when that score is consistent.

Test oracles. The oracle problem is the gap between running a program and knowing whether its output is right (Barr et al., 2015). Property-based, bounded-exhaustive and metamorphic testing need fewer expected outputs (Claessen & Hughes, 2000; Boyapati et al., 2002; Chen et al., 1998; 2018), mutation testing measures how well tests detect faults (Jia & Harman, 2011), and language models write tests and expected outputs (Kang et al., 2023; Schäfer et al., 2024). A bounded-exhaustive reference, exact on the inputs it can enumerate and silent elsewhere, is our reliable source.

3 Oracle Bias in Test-Based Selection

3.1 Preliminaries

An instance of the selection problem consists of a pool of n candidate programs $c_1, \dots, c_n$ and T test inputs $x_1, \dots, x_T$ drawn independently from an input generator. A candidate is correct if it returns the true output $y^\star(x)$ on every input x of the task. The oracle o maps an input to an expected output. The count of a candidate is $s(c) = \sum_{t=1}^{T} \mathbb{1}[c(x_t) = o(x_t)]$, and the count rule selects the candidate with the largest count, breaking ties at random. For two candidates c and c', the disagreement set is $D(c, c') = \{x : c(x) \neq c'(x)\}$; $\Pr(D)$ is the probability that a generated input falls in it and N_D the number of test inputs in it. For a pool, D denotes the inputs on which at least two candidates differ.

Proposition 1 (Pointwise cancellation). *For every oracle o and every pair of candidates c and c',*

$$s(c) - s(c') = \sum_{x_t \in D(c,c')} \Big(\mathbb{1}[c(x_t) = o(x_t)] - \mathbb{1}[c'(x_t) = o(x_t)] \Big). \tag{1}$$

Table 1: Oracles of five similarity levels on the same candidates and inputs (32B pools, six-benchmark means, %). μ, $p_\star$, p_{wrong} and $h = (\sqrt{p_\star} - \sqrt{p_{\text{wrong}}})^2$ refer to D; μ and h are fractions, computed from the listed $p_\star$ and p_{wrong}. ρ is the error agreement rate, measured on HumanEval+, LiveCodeBench and CodeContests; a is the overall accuracy. The last three columns give the accuracy of the count and of DisJudge and their difference. L5 is evaluated where the reference covers D. Per-benchmark values are in Appendix Table 21.

	Oracle	μ	ρ	$p_\star$	p_{wrong}	a	h	Count	DisJudge	Gain
L1	Same model, same prompt	+0.139	18.55	37.18	51.05	87.33	0.0110	54.40	60.04	+5.64
L2	Same model, other prompt	+0.085	13.72	40.33	48.78	89.02	0.0040	55.59	59.50	+3.91
L3	Larger model, same family	+0.022	5.73	41.78	44.00	88.67	0.0003	57.09	58.31	+1.22
L4	Other model family	−0.063	2.38	43.30	37.00	87.83	0.0025	58.62	59.10	+0.48
L5	Executable reference	−1.000	0.00	100.00	0.00	100.00	1	60.09	60.33	+0.24

Proposition 2 (Margin and overall accuracy). *Let $c_\star$ be correct, c' wrong and $D = D(c_\star, c')$. Define $p_\star = \Pr(o(x) = c_\star(x) \mid x \in D)$, $p_{\text{wrong}} = \Pr(o(x) = c'(x) \mid x \in D)$, $p_{\text{other}} = 1 - p_\star - p_{\text{wrong}}$ and the bias $\mu = p_{\text{wrong}} - p_\star$. Then*

$$\mathbb{E}\big[s(c_\star) - s(c')\big] = T \Pr(D)(p_\star - p_{\text{wrong}}) = -T \Pr(D)\, \mu. \tag{2}$$

The overall accuracy $a = \Pr(o(x) = y^\star(x))$ satisfies $a = \Pr(D)\, p_\star + (1 - \Pr(D))\, a_{\text{off}}$, where a_{off} is the accuracy of the oracle outside D, a term that the margin in Eq. (2) does not contain.

Corollary 1 (Overall accuracy cannot certify the ranking). *If $a > 1 - \Pr(D)/2$, every oracle with overall accuracy a has $\mu < 0$. If $a < 1 - \Pr(D)/2$, some oracle with this accuracy has $\mu > 0$.*

Remark. Proposition 1 holds for any oracle and any pair, so an answer placed outside D changes no pairwise margin, and a call spent there is wasted. Proposition 2 says that the count prefers the wrong candidate in expectation exactly when $\mu > 0$, and that a enters the margin only through $p_\star$. The disagreement set of a pair lies inside that of its pool, which holds 12.2% of the inputs on average in the 32B pools, so Corollary 1 asks for an overall accuracy of at least 93.9% before it can rule out a positive bias; our oracles reach 87% to 89%. All proofs are in Appendix B.

3.2 Bias on the Disagreement Set

We measure μ, $p_\star$ and p_{wrong} of Proposition 2 on six benchmarks with pools of $n = 20$ candidates and $T = 100$ generated inputs, whose candidates, inputs and expected outputs come from Qwen2.5-Coder-32B (Section 5.1). Fig. 1(a) shows the distribution of μ over 1,920 candidate pairs. Most pairs lie slightly below zero, where the oracle helps, but 31.0% have $\mu > 0$ and the tail reaches $\mu \approx 0.7$. Pooled over the disagreement inputs of each benchmark and averaged, the oracle returns the correct output on 37.2% of them and the output of the wrong candidate on 51.1% (Table 1). Fig. 1(b) traces the wrong selections of the count. We attribute a wrong selection in a pool that holds a correct candidate to bias on D when $\mu > 0$ for the selected candidate and a correct one, and to random errors or ties otherwise. On average 31.7% of the wrong selections are attributed to bias on D, 55.4% come from pools without a correct candidate, where no rule can succeed, and the rest from random errors and ties. Among the wrong selections in pools that do hold a correct candidate, 41.0% become correct when the answers of the oracle on D are replaced by the reference, mostly attributed to bias. Fig. 1(c) shows that D holds 12.2% of the inputs on average and the reference covers 83.3% of it.

Two further measurements separate the bias from the overall accuracy. First, two oracles whose overall accuracies are matched on each benchmark (87.2% on average) but differ on D lead to selection accuracies that differ by 8.4 points on average (Appendix Table 10). Second, adding tests does not remove the effect of the bias: from $T = 10$ to $T = 200$ the accuracy of the count rises by 0.3 to 0.4 points and stays 15 to 22 points below pass@20, the share of instances whose pool holds a correct candidate (Appendix Fig. 4). By Proposition 2, the margin of the count is set by $\Pr(D)$ and μ and does not involve a_{off}, so more tests only sharpen whatever preference the oracle already has.

3.3 Similarity Between Oracle and Generator

If the bias comes from errors that the oracle shares with the generator, it should shrink as the two share less. We keep every candidate and input fixed and change only the oracle, over five levels of similarity to the generator: L1, the generator with the prompt that produced the candidates; L2, the generator with another prompt; L3, a larger model of the same family; L4, a model of another

family; and L5, the bounded-exhaustive executable reference, which is exact where it applies. From L1 to L4 (Table 1), μ falls from $+0.139$ to -0.063, and the error agreement rate ρ, the share of wrong candidate outputs that the oracle reproduces exactly (Kim et al., 2025), falls from 18.6% to 2.4%. The overall accuracy varies only from 87.3% to 89.0% and does not follow the bias: L2 has the highest a but the second largest μ. By Proposition 2, a change in $p_\star$ moves a only by $\Pr(D)$ times that change, which is below one point here. Switching to a cross-family oracle also helps and raises the count from 54.4% to 58.6%, still below the 60.0% of DisJudge with the same-model oracle.

4 DisJudge: Calibrated Adjudication on the Disagreement Set

Section 3 suggests three design choices: ask only about inputs in D, ask first about the input that is most informative, and use every answer according to how reliable its source is. This section describes the three choices and analyzes the last one, which is where DisJudge departs from earlier rules.

4.1 Querying the Disagreement Set

DisJudge first executes every candidate on every test input and forms D, the inputs on which at least two candidates return different outputs. By Proposition 1, only these inputs can change the order of any two candidates, so reliable calls are made only on D. The next input is chosen by GBS (Dasgupta, 2004; Nowak, 2011). For each input $x \in D$ not yet queried, we group the candidates by their output on x and compute the share of the current weight held by the largest group, $g(x) = \max_v \sum_{c:\, c(x)=v} w_c / \sum_c w_c$, where w_c is the weight of candidate c defined below. We query the input with the smallest $g(x)$, whose answer moves the largest share of the weight, break ties by input order, and recompute g after every answer. The budget of m calls never exceeds N_D.

4.2 Calibrated Likelihood Update

Each answer comes from a source with a calibrated confusion matrix $P(y \mid z, q)$, the probability that the source returns y when the true output is z, for a query of type q; queries of one type share one calibrated matrix (Dawid & Skene, 1979). The oracle has the matrix P_o and the reliable source P_r. Candidate c predicts that the true output at x is $c(x)$, so an answer y at x supports c in proportion to $P(y \mid c(x), q_x)$. DisJudge starts from equal weights and scores each candidate by the calibrated log-likelihood of all answers on record on D,

$$\log w_c = \sum_{x \in D} \log P_o\big(o(x) \mid c(x), q_x\big) + \sum_{k=1}^{m} \log P_r\big(y_k \mid c(x_k), q_{x_k}\big), \tag{3}$$

where y_k is the answer of the k-th call, made at input x_k. We call Eq. (3) the soft update and $\log w_c$ the calibrated score of c. The selected candidate maximizes w_c, with ties broken by the count. Inputs outside D add the same term to every candidate and are left out. Every calibrated probability is kept positive, so no weight reaches zero and no candidate is removed; for the exact reference a small positive floor replaces its zero error, small enough that one reliable answer outweighs all answers of the oracle. The vote treats every answer as one more test; it is Eq. (3) when both sources have a symmetric confusion matrix with one common accuracy (Appendix B.7). Hard elimination removes every candidate that a reliable answer contradicts; as the error of the reliable source tends to zero, Eq. (3) puts the same candidates last and ranks the others by the calibrated answers of the oracle. Algorithm 1 in Appendix C.4 gives the procedure, and Appendix C.6 works one instance by hand.

4.3 Theoretical Analysis

Assumption 1 (Class-conditional answers)**.** For a pair $(c_\star, c')$, the answers of the oracle on $D(c_\star, c')$ are independent, and each equals $c_\star(x)$, $c'(x)$ or another output with probabilities $(p_\star, p_{\text{wrong}}, p_{\text{other}})$ when $c_\star$ is correct, and with the first two exchanged when c' is correct; Eq. (3) is built on them.

This is the class-conditional model of Dawid & Skene (1979): the answer depends on the true output and not on which program produced it. In a pool, Eq. (3) uses one matrix per query type, and the results apply to the pairs whose bias the calibration matches. Below, we use the Kullback–Leibler

divergence $K(p,q) = (p-q)\log(p/q)$ and the squared Hellinger distance $h(p,q) = (\sqrt{p}-\sqrt{q})^2$ between the two answer distributions of a pair (Kullback & Leibler, 1951; Cover & Thomas, 2006).

Theorem 1 (Count and calibrated score under a biased oracle)**.** *Under Assumption 1, let $M_T = s(c_\star) - s(c')$, and let $L_T = \log(w_{c_\star}/w_{c'})$ be computed by Eq. (3) from the answers of the oracle alone. (i) $M_T/T \to -\Pr(D)\,\mu$ almost surely, so $\Pr(M_T < 0) \to 1$ as $T \to \infty$ whenever $\mu > 0$. (ii) If $p_\star \neq p_{\text{wrong}}$ and $\Pr(D) > 0$, then $\mathbb{E}[L_T] = T\Pr(D)\,K(p_\star, p_{\text{wrong}}) > 0$ and*

$$\Pr(L_T \le 0) \;\le\; \big(1 - \Pr(D)\,h\big)^T \;\le\; e^{-T\Pr(D)\,h}, \qquad h = h(p_\star, p_{\text{wrong}}). \tag{4}$$

(iii) Adding m reliable answers on $D(c_\star, c')$ with error $\varepsilon < \frac{1}{2}$ adds $m\,K(1-\varepsilon,\varepsilon)$ to $\mathbb{E}[L_T]$ and multiplies Eq. (4) by $(1 - h(1-\varepsilon,\varepsilon))^m$, whereas under the vote they add $m(1-2\varepsilon)$ to $\mathbb{E}[M_T]$.

Remark. The count is consistent only when $\mu < 0$, and with a same-model oracle 31.0% of the pairs have $\mu > 0$. The calibrated score needs only $p_\star \neq p_{\text{wrong}}$, because a matching calibration turns even a biased oracle into evidence. For a single pair the score is $L_T = \log(p_\star/p_{\text{wrong}})\,M_T$, so it agrees with the count when $\mu < 0$ and reverses it when $\mu > 0$. For a fixed $p_\star + p_{\text{wrong}}$, the rate h vanishes at $\mu = 0$ and grows on both sides. Theorem 1 thus implies that the gain over the count can come only from the reliable answers when $\mu < 0$, and that the oracle adds the least evidence when it is almost unbiased; Section 5.4 compares both with the measurements, and Appendix B.8 shows that the sign of the calibrated weight is also hardest to estimate near $\mu = 0$.

Proposition 3 (Three update rules for a reliable answer)**.** *Let the count rank c' above $c_\star$ by $\Delta > 0$, and let each reliable call on $D(c_\star, c')$ return $c_\star(x)$ with probability $1-\varepsilon$ and $c'(x)$ otherwise, independently. (i) Under the vote the expected margin after m calls is $\Delta - m(1-2\varepsilon)$, so $m \ge \Delta/(1-2\varepsilon)$ calls are needed in expectation. (ii) Under hard elimination each call removes c' with probability $1-\varepsilon$, whatever Δ is, but $c_\star$ survives m calls with probability $(1-\varepsilon)^m$. (iii) Under the soft update every call adds $K(1-\varepsilon,\varepsilon)$ to the expected log-odds of $c_\star$ and removes nothing. With two outputs per query and error ε_k at the k-th call, the soft update ranks by $\sum_k \lambda_k \mathbb{1}[c(x_k) = y_k]$ with $\lambda_k = \log\frac{1-\varepsilon_k}{\varepsilon_k}$; this is the vote when ε_k is constant and hard elimination as $\varepsilon_k \to 0$.*

Remark. The vote has to pay for the whole margin, which by Proposition 2 grows with $T\Pr(D)\mu$. Hard elimination does not pay for the margin, but its mistakes are permanent: at $\varepsilon = 0.08$ and six calls it keeps a correct candidate with probability 0.61. The soft update gains $K(1-\varepsilon,\varepsilon)$ in log-odds per call and keeps every candidate. Among reliable answers alone, its advantage over the vote comes from the spread of λ_k across calls; on the full pools it also weighs the oracle's answers.

5 Experiments

5.1 Setup

Benchmarks and pools. We use HumanEval+ and MBPP+ (Chen et al., 2021; Austin et al., 2021; Liu et al., 2023), LiveCodeBench (Jain et al., 2025), BigCodeBench (Zhuo et al., 2025), CodeContests (Li et al., 2022) and APPS (Hendrycks et al., 2021). Candidate pools come from Qwen2.5-Coder-7B and 32B (Hui et al., 2024) and Qwen3-Coder-480B-A35B (Qwen Team, 2025a); five more generators appear in Appendix E.6. For each benchmark and generator we build $4{,}096$ instances; tasks are reused across instances, and each instance has its own pool of $n = 20$ candidates and $T = 100$ inputs written by the generator. A candidate counts as correct if it passes the ground-truth tests of its benchmark, and pass@20 bounds the accuracy of every rule from above. The default setting for detailed analyses is HumanEval+ with the 32B pool, and figures and appendix tables abbreviate the benchmarks as HE+, MBPP+, LCB, BCB, CC and APPS.

Oracle, reliable source and budget. The default oracle is L1, the generator with its own prompt. The reliable source is the bounded-exhaustive reference; it answers an input only when the input lies within its enumeration bound, and an abstention still costs a call. We ration its answers with a budget, to mimic the cost of a human or a slow checker. Every rule that queries the reliable source (✓ in Table 2) receives the same expected budget of 0.2 calls per disagreement input, 1.92 calls per instance in the default setting. The confusion matrices are calibrated on labeled disagreement inputs of each source (Appendix C.3). We report selection accuracy, the share of instances whose selected candidate is correct, and repeat each setting on three further pools (Appendix E).

Controlled simulation. To see how each update rule behaves under a noisy source, we also run the algorithm where every quantity is controlled: 256 instances per benchmark, four candidates of

Table 2: Selection accuracy (%) of 26 rules on six benchmarks with candidate pools from three generators (7B and 32B: Qwen2.5-Coder; 480B: Qwen3-Coder-480B-A35B), and the mean over the 18 settings (Avg). All rules run on the same 4,096 instances per setting. Rules marked ✓ query the reliable source at the same expected budget of 0.2 calls per disagreement input. Grey rows are reference points: random choice and the pass@20 upper bound. Among the other rows, the best is in bold and the second best underlined.

		HumanEval+			MBPP+			LiveCodeBench			BigCodeBench			CodeContests			APPS			
Rule	✓	7B	32B	480B	7B	32B	480B	7B	32B	480B	7B	32B	480B	7B	32B	480B	7B	32B	480B	Avg ↑
Random choice		46.5	37.3	34.3	39.7	44.4	47.0	27.3	25.4	34.7	26.6	32.6	31.9	19.2	24.5	34.7	19.1	23.1	27.3	32.0
Upper bound (pass@20)		87.7	90.2	98.8	85.6	85.1	92.5	59.0	62.7	70.5	64.7	77.2	72.7	57.8	60.8	63.1	57.6	63.3	70.5	73.3
Count of passed tests		71.8	69.9	76.7	64.3	70.1	72.6	45.1	46.4	50.3	48.7	55.6	56.2	39.5	39.4	42.9	40.7	45.1	50.4	54.8
CodeT		71.4	71.5	78.2	67.7	73.7	72.2	47.1	47.8	53.0	50.0	60.1	57.9	39.5	40.7	46.7	43.7	50.0	54.5	57.0
AlphaCode clustering		73.2	**75.5**	79.9	66.9	72.0	76.8	49.2	49.4	53.6	51.3	57.1	60.0	39.9	40.3	46.1	42.5	45.8	54.7	57.5
MBR-Exec		72.2	72.4	77.6	67.1	72.8	74.4	46.1	49.2	53.3	50.1	59.5	56.5	39.9	41.2	46.1	44.6	46.8	52.2	56.8
Execution self-consistency		72.4	73.2	79.7	66.7	70.5	73.9	47.7	47.1	51.8	48.7	56.8	56.4	41.3	40.0	45.8	43.1	48.0	52.4	56.4
LEVER		76.5	69.8	76.9	70.0	74.5	74.3	48.7	49.3	53.3	49.6	58.5	59.2	42.2	42.8	45.9	44.0	48.3	49.4	57.4
Coder-Reviewer reranking		74.1	73.9	81.9	69.3	71.7	75.5	48.5	50.0	51.7	52.8	55.9	58.7	42.3	40.7	46.2	43.8	48.5	51.9	57.6
Outcome/process reward model		72.3	71.7	79.5	68.5	71.5	75.7	45.3	50.6	52.3	52.1	58.6	58.7	40.7	39.0	46.5	45.3	47.5	51.8	57.1
Fault-aware code ranker		74.8	69.2	78.5	69.5	71.5	74.4	47.3	51.0	51.6	48.6	59.0	58.8	43.3	43.0	45.9	46.2	47.4	52.1	57.3
LLM judge, pairwise	✓	75.6	71.4	78.3	67.1	73.1	75.8	45.5	48.7	51.2	50.5	56.9	55.7	42.2	41.6	47.1	44.6	50.6	53.0	57.2
Tournament elimination	✓	72.1	71.1	75.7	66.8	73.1	77.0	48.4	48.7	53.6	54.1	59.4	58.6	39.8	42.7	45.7	44.9	45.9	51.8	57.2
Self-critique scoring		74.2	73.4	78.0	65.3	74.1	76.5	45.8	49.8	50.5	50.5	58.0	60.6	40.1	42.8	45.8	43.8	47.0	51.4	57.1
More generated tests	✓	75.6	74.2	78.2	66.7	69.9	74.6	46.7	51.4	51.9	50.4	56.4	56.9	43.5	40.6	44.5	42.5	48.4	52.7	56.9
Test-designer agent	✓	77.1	72.7	80.7	65.8	72.0	76.5	45.1	46.4	53.2	50.4	58.3	59.4	44.5	42.1	47.0	42.8	48.8	53.5	57.6
Self-Debugging		77.3	75.0	75.9	67.1	71.9	75.3	48.0	47.6	53.8	51.6	59.9	61.1	42.6	43.0	43.9	41.7	50.2	50.6	57.6
S*		75.9	70.5	78.2	68.1	71.9	74.7	46.4	47.0	50.3	50.8	56.3	60.2	41.3	42.8	48.2	44.0	49.9	52.2	57.2
Distinguishing inputs	✓	73.3	71.4	80.9	67.9	75.1	74.3	48.2	47.8	53.4	54.6	56.4	58.2	41.0	40.3	46.6	44.6	47.4	52.9	57.5
Query by committee	✓	71.8	72.7	77.1	66.9	**75.9**	76.4	46.0	48.0	53.0	50.7	57.2	58.1	44.4	44.1	45.2	42.4	46.4	50.9	57.1
Noisy GBS	✓	73.3	71.9	80.9	65.5	71.7	73.0	47.5	49.4	52.9	50.6	58.1	58.7	42.4	41.1	44.8	44.4	48.7	54.6	57.2
Margin-ordered queries	✓	72.2	73.5	76.9	68.9	70.7	75.4	47.0	47.4	52.5	50.9	59.3	58.9	40.5	41.4	46.2	43.5	47.5	50.8	56.9
Top-2 queries	✓	72.2	69.8	79.9	68.3	74.5	74.7	47.8	49.6	53.1	51.2	56.6	59.4	41.9	40.8	45.7	44.8	49.6	52.1	57.3
Random queries within D	✓	72.4	73.2	79.6	65.1	73.3	74.0	47.2	49.1	50.5	49.8	59.4	57.9	41.1	40.1	42.5	44.0	46.3	52.2	56.5
DisJudge, vote update	✓	75.1	73.2	79.6	67.7	72.9	75.5	48.5	49.9	53.2	52.5	59.0	58.6	42.8	41.9	46.5	43.8	48.1	53.4	57.9
DisJudge, hard elimination	✓	75.4	73.5	80.0	67.7	72.9	76.0	49.2	50.3	53.3	53.0	58.9	58.9	42.6	42.5	46.6	44.3	48.6	53.7	58.2
DisJudge, soft update	✓	**77.4**	75.4	**82.4**	**70.2**	75.6	**78.3**	**51.1**	**52.4**	**55.3**	**55.2**	**61.2**	**61.3**	**45.6**	**44.2**	**49.1**	**46.6**	**51.5**	**55.3**	**60.5**

which one is correct, twelve inputs, a source with errors independent of those of the candidates and a mean error per query of $\bar{\varepsilon} = 0.08$, six requested calls, and error rates estimated from calibration records (Appendix C.5). Without any answer, selection is at chance (24.0%). Its accuracies are not comparable with those on the full pools, and we use it mainly to compare the parts of the method.

5.2 Main Results

In the main comparison of Table 2, DisJudge with the soft update has the best mean accuracy, 60.5%, which is 5.7 points above the count and 2.8 points above the best baseline rule. It is first in 16 of the 18 settings and second in the other two, where AlphaCode clustering and query by committee lead by 0.1 and 0.3 points. The gain over the count lies between 4.8 and 6.5 points across the 18 settings (Appendix Table 14). The vote and hard elimination query D in the same way and reach 57.9% and 58.2%. With the exact reference, hard elimination and the soft update both put the candidates that agree with every reliable answer first; the soft update ranks them by the calibrated answers of the oracle instead of the count, which gives it 2.3 points. The other rules that query the reliable source, including the three from active learning, reach 56.5% to 57.6%.

Safety and cost. By design, the soft update removes no candidate. Hard elimination keeps every correct candidate because the reference is exact where it answers, while the rules that eliminate by their own criteria lose 2.0% to 2.7% of the correct candidates in the default setting (Appendix Table 13). Our executable reference takes 11.0 ms per instance, or 7.9% of the 138.6 ms in total.

Budget, generality and composition. The soft update leads the other four rules of the budget sweep by 0.34 to 2.59 points on HumanEval+ from 0.02 to 1 call per disagreement input, and by 1.7 to 3.8 points per benchmark at 0.2 and 1; a fifth of D gives 75.4% in the default setting, against 77.3% for all of D (Appendix E.5). Across eight generators, three input generators, four pool sizes and four task families, the four-benchmark mean of DisJudge ranges from 57.3% to 62.0%, and placing it after four first-stage rules improves each of them (Appendix E.6 and E.7).

Table 3: Adding the parts of DisJudge. Left: controlled simulation, five query rules crossed with three update rules (Vote: vote update; Hard: hard elimination; Soft: soft update), six-benchmark means in %, six calls and $\bar{\varepsilon} = 0.08$; 24.0% without any answer. Right: full pools of the 32B generator, six-benchmark means in %. In rows two and three on the right, a reliable answer replaces the oracle's output on the queried input; best in bold.

Query rule	Vote	Hard	Soft ↑
Random over all inputs	78.19	53.45	78.58
Random within D	86.26	56.64	86.78
GBS order within D (ours)	92.32	58.07	93.23
Margin order within D	90.62	57.94	90.95
Top-2 order within D	92.58	58.20	93.29

Full pools	Avg ↑
Count, no reliable calls	54.40
Count, calls in D, random order	55.53
Count, calls in D, GBS order	57.72
Vote update	57.52
Hard elimination	57.79
Soft update (DisJudge)	**60.04**

Table 4: Varying the source and the budget in the controlled simulation (six-benchmark means, %). Default: independent errors, $\bar{\varepsilon} = 0.08$ and six calls. Vote, Hard and Soft as in Table 3; Kept and Emptied: shares of correct candidates that hard elimination keeps and of pools it empties. The correlated source has errors correlated with those of the candidates and is weaker overall and on D; the reference answers 80% of the queries. Best update rule per row in bold. Last row: pools without a correct candidate, where Kept is undefined.

Change	Value	Vote	Hard	Soft ↑	Kept ↑	Emptied ↓
Default	$\bar{\varepsilon} = 0.08$, 6 calls	92.32	58.07	**93.23**	58.59	38.74
Source error	$\bar{\varepsilon} = 0.02$	97.72	86.39	**98.11**	87.30	11.98
	$\bar{\varepsilon} = 0.20$	72.33	24.61	**76.56**	24.93	71.22
Requested calls	2	81.12	61.25	**81.18**	86.50	2.40
	12	93.62	47.66	**94.21**	47.92	50.71
Correlated source	1 call	26.63	26.63	**38.02**	56.58	0.00
	6 calls	28.45	9.70	**68.75**	10.03	60.35
	12 calls	26.50	3.71	**69.47**	4.04	79.04
Exact reference	4.80 calls	84.05	84.05	84.05	100.00	0.00
No correct candidate		0.00	0.00	0.00	–	92.38

5.3 Ablation Studies

We add the parts of DisJudge in turn (Table 3) and then vary the source and the budget (Table 4).

Only calls in D help. Moving random calls from all inputs into D raises the accuracy of the soft update from 78.6% to 86.8% in the simulation. On the full pools, calls inside D raise the count from 54.40% to 55.53% in random order, since only answers on D can change a margin (Proposition 1).

Ordering the calls by GBS helps. The GBS order adds 6.4 points to the soft update over a random order and 2.3 points over the margin order, and the top-2 order, which asks only between the two leaders, does about as well on pools of four. On the full pools it adds 2.2 points. Recomputing the order after every answer adds 4.7 points, and computing g from the outputs beats predicting it with a model by 2.5 points at 0.09 instead of 689 ms per instance (Appendix F.3).

The soft update is the most accurate at all three source errors and removes nothing. It beats the vote by 0.4, 0.9 and 4.2 points at $\bar{\varepsilon} = 0.02$, 0.08 and 0.2, a gap that by Proposition 3 can come only from the spread of λ_k across calls. Hard elimination needs an almost exact source, since at $\bar{\varepsilon} = 0.2$ it keeps only a quarter of the correct candidates and empties 71% of pools.

More calls help the soft update and hurt hard elimination. From two to twelve requested calls, the soft update rises from 81.2% to 94.2% and the vote to 93.6%, while hard elimination peaks at two calls and falls to 47.7%, because each further call may remove the correct candidate permanently.

Calibration helps most when the source is weak on D. A source with errors correlated with those of the candidates is weaker overall and much weaker on D (68.6% against 91.6%). Under this source the vote remains at 26.5% to 29.5% for any number of calls and hard elimination falls to 3.7%, while the soft update climbs from 38.0% at one call to 69.5% at twelve calls, since the calibrated weights discount the queries where the source fails. With the exact reference as the source, the three update rules select alike (84.05%), as Proposition 3 gives for $\varepsilon \to 0$. For the default source, 64 calibration records per query type already suffice to match the true error rates (Appendix Table 26).

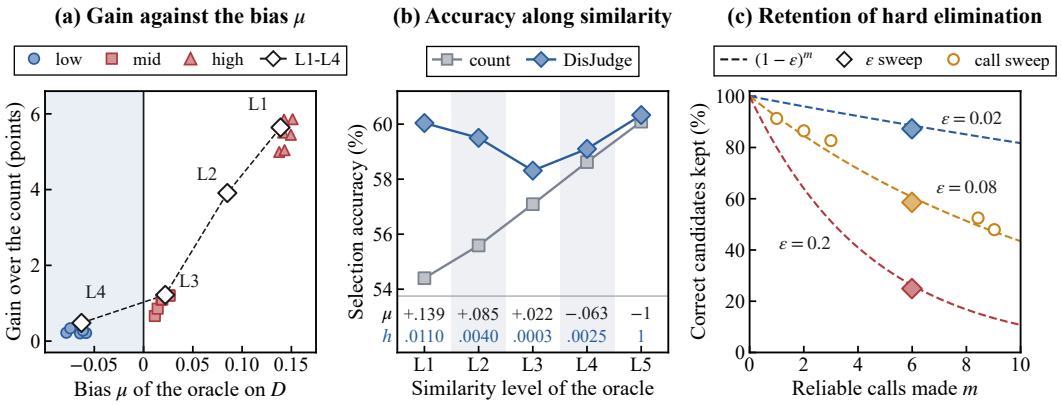

Figure 2: Three consequences of Section 4.3 against the measurements. (a) Gain of DisJudge over the count against the bias μ of the oracle (shaded: $\mu < 0$): six benchmarks at a low, mid and high bias level and the similarity levels L1 to L4 of Table 1. (b) Selection accuracy of the count and of DisJudge along the similarity levels, with μ and $h = (\sqrt{p_\star} - \sqrt{p_{\text{wrong}}})^2$ of each level; L5 is evaluated where the reference covers D. (c) Share of correct candidates kept by hard elimination in the simulation against the calls made m: diamonds for three error rates at six calls, circles for one to twelve requested calls at $\bar{\varepsilon} = 0.08$, and dashed curves for $(1-\varepsilon)^m$ with $\varepsilon = \bar{\varepsilon}$.

Failure cases. On pools without a correct candidate, every rule scores zero and hard elimination empties 92.4% of them. The reference misses overflow and complexity errors, which lie beyond its bound, and there the accuracy stays at 23.3% and 27.1% at every coverage (Appendix F.5).

5.4 Comparison with the Theory

We close by comparing three consequences of Section 4.3 with the measurements (Fig. 2). First, Theorem 1 implies that when $\mu < 0$ the calibrated score ranks each pair like the count, so the gain can come only from the reliable answers, while for $\mu > 0$ the two rankings are opposite. In a sweep over three bias levels of the oracle, the gain is 0.2 to 0.5 points where $\mu \approx -0.07$, 0.7 to 1.2 points where $\mu \approx +0.02$ and 5.0 to 5.9 points where $\mu \approx +0.14$, and the similarity levels L1 to L4 follow the same trend (panel a). Second, the count improves steadily from L1 to L5 as μ falls, while DisJudge largely follows h. Among the four model-written oracles, it is most accurate at L1, where h is largest, and least accurate at L3, where $p_\star \approx p_{\text{wrong}}$ and h is almost zero, so on these pools the L1 oracle helps the calibrated score more than the L3 oracle (panel b and Appendix Table 21). With the executable reference, the count and DisJudge nearly meet, at 60.1% and 60.3%. Third, Proposition 3 gives $(1-\varepsilon)^m$ as the share of correct candidates that hard elimination keeps for a constant error. In the simulation, where the error varies across queries around $\bar{\varepsilon}$, the retention lies within 2.1 points of this curve across the three error rates and within 5 points across the number of calls (panel c).

6 Conclusion

We studied test-based selection of generated programs when the expected outputs come from a model. The count compares two candidates only where they disagree, and there an oracle that shares the errors of the generator can favor the wrong program while remaining accurate overall. We proved that on such a pair the count selects the wrong program with probability tending to one, while a calibrated likelihood score picks the correct one at a rate that grows with the Hellinger distance between the two answer distributions. DisJudge spends a few reliable calls on the disagreement set and ranks the candidates by the calibrated likelihood of all answers. It raises the mean selection accuracy from 54.8% to 60.5% over 18 settings, and its gain follows the bias of the oracle. It keeps a similar accuracy across eight generators and three input generators and improves four first-stage rules.

Limitations. DisJudge needs a reliable source and a calibrated confusion matrix for each source. Our reliable source is a bounded-exhaustive reference that covers 83% of the disagreement inputs, and other sources, such as human answers, may be slower or noisier and need their own calibration.

Reproducibility Statement

The method is specified by Eq. (3) and Algorithm 1. Appendix B proves every statement of Section 3 and Section 4.3. Appendix C describes the candidate pools, the oracles, the reliable source, the calibration of the confusion matrices, the controlled simulation and the other selection rules, and works one instance of the algorithm by hand. Appendices D to G report the numbers behind the figures and tables of the main text, with per-benchmark values and the denominators of all rates. The candidate pools, the simulation and the evaluation code will be released upon publication.

AI Use Statement

Generative AI tools were used only to edit the wording of text written by the authors, to search for related work and check bibliographic details, and to write code for plots, tables and typesetting. They were not used for any task that the ICLR 2027 AI policy requires authors to disclose: developing theory, formulating or proving mathematical claims, proposing or refining hypotheses, designing, implementing or running the method or experiments, generating, cleaning or analyzing data, translating text, or interpreting results. The models in our experiments are objects of study, not writing aids. The authors reviewed all AI-assisted work and take full responsibility for the content of this paper.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, 2015. doi: 10.1109/TSE.2014.2372785.

Chandrasekhar Boyapati, Sarfraz Khurshid, and Darko Marinov. Korat: Automated testing based on Java predicates. In *Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pp. 123–133, 2002. doi: 10.1145/566172.566191.

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.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. CodeT: Code generation with generated tests. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=ktrw68Cmu9c.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Tsong Yueh Chen, Shing Chi Cheung, and Siu Ming Yiu. Metamorphic testing: A new approach for generating next test cases. Technical Report HKUST-CS98-01, Department of Computer Science, The Hong Kong University of Science and Technology, 1998. Also available as arXiv:2002.12543.

Tsong Yueh Chen, Fei-Ching Kuo, Huai Liu, Pak-Lok Poon, Dave Towey, T. H. Tse, and Zhi Quan Zhou. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys*, 51(1):4:1–4:27, 2018. doi: 10.1145/3143561.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=KuPixIqPiq.

Koen Claessen and John Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP)*, pp. 268–279, 2000. doi: 10.1145/351240.351266.

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

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994. doi: 10.1007/BF00993277.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, Hoboken, NJ, 2nd edition, 2006. doi: 10.1002/047174882X.

Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, 2004. URL https://proceedings.neurips.cc/paper_files/paper/2004/hash/c61fbef63df5ff317aecdc3670094472-Abstract.html.

A. P. Dawid and A. M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 28(1):20–28, 1979. doi: 10.2307/2346806.

Sarah Fakhoury, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, and Shuvendu K. Lahiri. LLM-based test-driven interactive code generation: User study and empirical evaluation. *IEEE Transactions on Software Engineering*, 50(9):2254–2268, 2024. doi: 10.1109/TSE.2024.3428972.

Shashwat Goel, Joschka Strüber, Ilze Amanda Auzina, Karuna K. Chandra, Ponnurangam Kumaraguru, Douwe Kiela, Ameya Prabhu, Matthias Bethge, and Jonas Geiping. Great models think alike and this undermines AI oversight. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19621–19678, 2025. URL https://proceedings.mlr.press/v267/goel25b.html.

Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal Bayesian active learning with noisy observations. In *Advances in Neural Information Processing Systems (NIPS)*, volume 23, 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/hash/1e6e0a04d20f50967c64dac2d639a577-Abstract.html.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Bernard Grofman, Guillermo Owen, and Scott L. Feld. Thirteen theorems in search of the truth. *Theory and Decision*, 15(3):261–278, 1983. doi: 10.1007/BF00125672.

Steve Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2–3):131–309, 2014. doi: 10.1561/2200000037.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/c24cd76e1ce41366a4bbe8a49b02a028-Abstract-round2.html.

Dong Huang, Jie M. Zhang, Michael Luck, Qingwen Bu, Yuhao Qing, and Heming Cui. AgentCoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=IkmD3fKBPQ.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Jeevana Priya Inala, Chenglong Wang, Mei Yang, Andres Codas, Mark Encarnación, Shuvendu K. Lahiri, Madanlal Musuvathi, and Jianfeng Gao. Fault-aware neural code rankers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 13419–13432, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/5762c579d09811b7639be2389b3d07be-Abstract-Conference.html.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, et al. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Susmit Jha, Sumit Gulwani, Sanjit A. Seshia, and Ashish Tiwari. Oracle-guided component-based program synthesis. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE)*, volume 1, pp. 215–224, 2010. doi: 10.1145/1806799.1806833.

Ruyi Ji, Jingjing Liang, Yingfei Xiong, Lu Zhang, and Zhenjiang Hu. Question selection for interactive program synthesis. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 1143–1158, 2020. doi: 10.1145/3385412.3386025.

Yue Jia and Mark Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011. doi: 10.1109/TSE.2010.62.

Sungmin Kang, Juyeon Yoon, and Shin Yoo. Large language models are few-shot testers: Exploring LLM-based general bug reproduction. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE)*, pp. 2312–2323, 2023. doi: 10.1109/ICSE48619.2023.00194.

Elliot Myunghoon Kim, Avi Garg, Kenny Peng, and Nikhil Garg. Correlated errors in large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 30038–30066, 2025. URL https://proceedings.mlr.press/v267/kim25e.html.

S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. doi: 10.1214/aoms/1177729694.

Shuvendu K. Lahiri, Sarah Fakhoury, Aaditya Naik, Georgios Sakkas, Saikat Chakraborty, Madanlal Musuvathi, Piali Choudhury, Curtis von Veh, Jeevana Priya Inala, Chenglong Wang, and Jianfeng Gao. Interactive code generation via test-driven user-intent formalization. *arXiv preprint arXiv:2208.05950*, 2022.

Dacheng Li, Shiyi Cao, Chengkun Cao, Xiuyu Li, Shangyin Tan, Kurt Keutzer, Jiarong Xing, Joseph E. Gonzalez, and Ion Stoica. S*: Test time scaling for code generation. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 15964–15978, Suzhou, China, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-emnlp.865. URL https://aclanthology.org/2025.findings-emnlp.865/.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, et al. Competition-level code generation with AlphaCode. *Science*, 378 (6624):1092–1097, 2022. doi: 10.1126/science.abq1158.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, et al. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 21558–21572, 2023. URL https://openreview.net/forum?id=1qvx610Cu7.

Mistral AI. Introducing: Devstral 2 and Mistral Vibe CLI. https://mistral.ai/news/devstral-2-vibe-cli, December 2025. Blog post, December 9, 2025. Announces Devstral 2 (123B) and Devstral Small 2 (24B); open weights at https://

huggingface.co/mistralai/Devstral-2-123B-Instruct-2512 and https://huggingface.co/mistralai/Devstral-Small-2-24B-Instruct-2512.

Ansong Ni, Srini Iyer, Dragomir Radev, Veselin Stoyanov, Wen-tau Yih, Sida Wang, and Xi Victoria Lin. LEVER: Learning to verify language-to-code generation with execution. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26106–26128, 2023. URL https://proceedings.mlr.press/v202/ni23b.html.

Shmuel Nitzan and Jacob Paroush. Optimal decision rules in uncertain dichotomous choice situations. *International Economic Review*, 23(2):289–297, 1982. doi: 10.2307/2526438.

Robert D. Nowak. Noisy generalized binary search. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, 2009. URL https://proceedings.neurips.cc/paper_files/paper/2009/hash/556f391937dfd4398cbac35e050a2177-Abstract.html.

Robert D. Nowak. The geometry of generalized binary search. *IEEE Transactions on Information Theory*, 57(12):7893–7906, 2011. doi: 10.1109/TIT.2011.2169298.

Arjun Panickssery, Samuel R. Bowman, and Shi Feng. LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 68772–68802, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7f1f0218e45f5414c79c0679633e47bc-Abstract-Conference.html.

Qwen Team. Qwen3-Coder: Agentic coding in the world. https://qwenlm.github.io/blog/qwen3-coder/, July 2025a. Blog post, July 22, 2025.

Qwen Team. QwQ-32B: Embracing the power of reinforcement learning. https://qwenlm.github.io/blog/qwq-32b/, March 2025b. Blog post, March 6, 2025.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11:1297–1322, 2010.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering*, 50(1):85–105, 2024. doi: 10.1109/TSE.2023.3334955.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL https://minds.wisconsin.edu/handle/1793/60660.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory (COLT)*, pp. 287–294, 1992. doi: 10.1145/130385.130417.

Freda Shi, Daniel Fried, Marjan Ghazvininejad, Luke Zettlemoyer, and Sida I. Wang. Natural language to code translation with execution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3533–3546, 2022. doi: 10.18653/v1/2022.emnlp-main.231. URL https://aclanthology.org/2022.emnlp-main.231/.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-preference bias in LLM-as-a-judge. *arXiv preprint arXiv:2410.21819*, 2024.

Tianyi Zhang, Tao Yu, Tatsunori B. Hashimoto, Mike Lewis, Wen-tau Yih, Daniel Fried, and Sida I. Wang. Coder reviewer reranking for code generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 41832–41846, 2023. URL https://proceedings.mlr.press/v202/zhang23av.html.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, pp. 46595–46623, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=YrycTjllL0.

APPENDIX

A Notation **16**

B Proofs and Derivations **16**

B.1 Proof of Proposition 1 . . . 16
B.2 Proof of Proposition 2 and Corollary 1 . . . 16
B.3 The Distribution of the Margin . . . 18
B.4 Proof of Theorem 1 . . . 18
B.5 From a Pair to the Pool . . . 19
B.6 Proof of Proposition 3 . . . 19
B.7 The Multi-Valued Update and the Count . . . 20
B.8 Estimating the Weight of the Oracle . . . 20

C Implementation Details **20**

C.1 Candidate Pools and Oracles . . . 20
C.2 Reliable Source and Budget . . . 21
C.3 Calibration of the Confusion Matrices . . . 21
C.4 The Algorithm . . . 21
C.5 Controlled Simulation . . . 21
C.6 A Worked Example . . . 22
C.7 Other Selection Rules . . . 23

D Additional Diagnosis Results **24**

D.1 Number of Tests and the Margin . . . 24
D.2 Error Breakdown per Benchmark . . . 24
D.3 Overall Accuracy and the Accuracy Outside D . . . 25

E Additional Results on the Candidate Pools **25**

E.1 All Rules in the Default Setting . . . 25
E.2 Safety and Cost . . . 25
E.3 Gains in All Settings . . . 25
E.4 Statistics of the Pools . . . 26
E.5 Budget . . . 26
E.6 Generality . . . 28
E.7 Composition with First-Stage Rules . . . 28
E.8 Similarity Levels per Benchmark . . . 28

F Complete Ablation Studies **29**

F.1 Update Rule and Source Error . . . 29
F.2 Query Rule and Update Rule . . . 29
F.3 Budget, Calibration Size and Query Order . . . 29
F.4 First-Stage Rules and Simulated Sources . . . 30

F.5 Reference Coverage and Pools Without a Correct Candidate 31
F.6 Per-Benchmark Gains and Calls to a Target 31

G Denominators of the Rates **32**

A Notation

Table 5 lists the symbols and where each is defined. Percentages in the tables are shares of instances unless a caption says otherwise, and a difference between two percentages is given in points.

B Proofs and Derivations

This appendix proves the statements of Sections 3 and 4.3. It also derives the distribution of the margin, extends Theorem 1 from a pair to a whole pool, and gives the soft update for outputs with more than two values. Throughout, the T test inputs are independent draws from the input generator.

B.1 Proof of Proposition 1

Write each count as a sum over the T inputs, subtract the two sums term by term, and split the inputs by membership in $D(c, c')$:

$$\begin{aligned} s(c) - s(c') &= \sum_{t=1}^{T} \Big(\mathbb{1}[c(x_t) = o(x_t)] - \mathbb{1}[c'(x_t) = o(x_t)] \Big) \\ &= \sum_{x_t \notin D(c,c')} \Big(\mathbb{1}[c(x_t) = o(x_t)] - \mathbb{1}[c'(x_t) = o(x_t)] \Big) \\ &\quad + \sum_{x_t \in D(c,c')} \Big(\mathbb{1}[c(x_t) = o(x_t)] - \mathbb{1}[c'(x_t) = o(x_t)] \Big). \end{aligned} \tag{5}$$

On an input outside $D(c, c')$ the two candidates return the same output, so $c(x_t) = o(x_t)$ holds exactly when $c'(x_t) = o(x_t)$ holds. The two indicators are then equal and the term vanishes, whatever the oracle returns. The first sum is therefore zero, and Eq. (1) follows from the second. □

B.2 Proof of Proposition 2 and Corollary 1

The margin. Apply Eq. (1) to the pair $(c_\star, c')$ and take the expectation over the independent inputs. Each term is nonzero only on the event $x_t \in D$, and on this event the correct output is $c_\star(x_t)$:

$$\begin{aligned} \mathbb{E}\big[s(c_\star) - s(c')\big] &= \sum_{t=1}^{T} \mathbb{E}\Big[\mathbb{1}[x_t \in D]\big(\mathbb{1}[o(x_t) = c_\star(x_t)] - \mathbb{1}[o(x_t) = c'(x_t)]\big)\Big] \\ &= \sum_{t=1}^{T} \Pr(D)\Big(\Pr\big(o(x) = c_\star(x) \mid x \in D\big) - \Pr\big(o(x) = c'(x) \mid x \in D\big)\Big) \\ &= T \Pr(D)\,(p_\star - p_{\text{wrong}}) \\ &= -T \Pr(D)\,\mu. \end{aligned} \tag{6}$$

The overall accuracy. The oracle is right at x exactly when $o(x) = y^\star(x)$, and $y^\star(x) = c_\star(x)$ on every input because $c_\star$ is correct. Conditioning on membership in D,

$$\begin{aligned} a = \Pr\big(o(x) = y^\star(x)\big) &= \Pr(D)\,\Pr\big(o(x) = c_\star(x) \mid x \in D\big) \\ &\quad + \big(1 - \Pr(D)\big)\,\Pr\big(o(x) = y^\star(x) \mid x \notin D\big) \\ &= \Pr(D)\,p_\star + \big(1 - \Pr(D)\big)\,a_{\text{off}}. \end{aligned} \tag{7}$$

Since a_{off} does not appear in Eq. (2), the margin depends on the overall accuracy only through $p_\star$. □

Table 5: Symbols and terms.

Symbol or term	Meaning	Where
$c_1, \ldots, c_n$; n	Candidate programs of one instance and their number ($n = 20$ by default).	§3.1
$x_1, \ldots, x_T$; T	Test inputs drawn from the input generator and their number ($T = 100$).	§3.1
$y^\star(x)$	True output at input x.	§3.1
$c_\star$, c'	A correct candidate and a wrong candidate.	§3.1
o	Oracle: the source of the expected outputs of the tests.	§3.1
$s(c)$	Count: number of tests on which $c(x_t) = o(x_t)$.	§3.1
$\mathbb{1}[\cdot]$	Indicator: one when the condition holds and zero otherwise.	§3.1
$D(c, c')$, D	Disagreement set of a pair; for a pool, the inputs on which at least two candidates differ.	§3.1
$\Pr(D)$, N_D	Probability that a generated input falls in D; number of test inputs in D.	§3.1
$p_\star$, p_{wrong}, p_{other}	Probabilities that the oracle returns $c_\star(x)$, $c'(x)$ or another output, for $x \in D$.	Prop. 2
μ	Bias of the oracle on D, $\mu = p_{\text{wrong}} - p_\star$.	Prop. 2
a, a_{off}	Overall accuracy of the oracle and its accuracy outside D.	Prop. 2
L1 to L5	Similarity levels of the oracle, from the generator with its own prompt (L1) to the executable reference (L5).	§3.3
ρ	Error agreement rate: share of wrong candidate outputs that the oracle reproduces exactly, following Kim et al. (2025).	§3.3
$P(y \mid z, q)$; P_o, P_r	Calibrated confusion matrix of a source for query type q; of the oracle and of the reliable source.	§4.2
q, q_x	Query type, and the type of input x; queries of one type share one calibrated confusion matrix.	§4.2
w_c	Weight of candidate c under the soft update.	§4.2
$g(x)$	Share of the current weight held by the largest group of candidates with one output v at x (generalized binary search score).	§4.1
m; x_k, y_k	Number of reliable calls per instance; input and answer of the k-th call.	§4.1, 4.2
ε, ε_k, $\bar{\varepsilon}$	Error rate of a reliable source; its value at the k-th call; its mean over calls.	Thm. 1
λ_k	Log-odds weight $\log \frac{1-\varepsilon_k}{\varepsilon_k}$ of the answer to the k-th call.	Prop. 3
λ, λ_r	Weight $\log(p_\star / p_{\text{wrong}})$ of an agreement with the oracle; weight $\log \frac{1-\varepsilon}{\varepsilon}$ of a reliable answer.	App. B.4
M_T, L_T	Count difference $s(c_\star) - s(c')$ and log-weight difference $\log(w_{c_\star} / w_{c'})$ after T tests.	Thm. 1
$K(p, q)$	Kullback–Leibler divergence $(p - q)\log(p/q)$ between the answer distributions of two hypotheses.	§4.3
$h(p, q)$, h	Squared Hellinger distance $(\sqrt{p} - \sqrt{q})^2$ between the same distributions; $h = h(p_\star, p_{\text{wrong}})$.	§4.3
Δ	Margin by which the count ranks c' above $c_\star$.	Prop. 3
Vote, hard, soft	The three update rules: one more test per answer, removal of contradicted candidates, and Eq. (3).	§4.2
Kept, emptied	Share of correct candidates that a rule keeps; share of pools it empties.	§5.3
Default setting	HumanEval+ with the Qwen2.5-Coder-32B pool.	§5.1
pass@20	Share of instances whose pool holds a correct candidate; an upper bound for every rule.	§5.1
S1 to S5	Simulated sources of the controlled simulation.	App. C.5
↑, ↓	Higher or lower is better; a shaded row marks DisJudge or its default setting.	Tables

The threshold. Because $a_{\text{off}} \leq 1$, Eq. (7) gives $p_\star \geq 1 - (1 - a)/\Pr(D)$. If $a > 1 - \Pr(D)/2$, this bound gives $p_\star > \frac{1}{2}$, and then $p_{\text{wrong}} \leq 1 - p_\star < \frac{1}{2} < p_\star$, so $\mu < 0$. For the converse, let $a < 1 - \Pr(D)/2$. If $a \geq 1 - \Pr(D)$, the oracle with $a_{\text{off}} = 1$, $p_\star = 1 - (1 - a)/\Pr(D) < \frac{1}{2}$ and $p_{\text{wrong}} = 1 - p_\star$ has overall accuracy a and $\mu = 1 - 2p_\star > 0$. If $a < 1 - \Pr(D)$, the oracle with $p_\star = 0$, $p_{\text{wrong}} = 1$ and $a_{\text{off}} = a/(1 - \Pr(D)) < 1$ has overall accuracy a and $\mu = 1$. With the average $\Pr(D) = 0.1222$ of the 32B pools the threshold is $1 - 0.0611 = 0.9389$. □

B.3 The Distribution of the Margin

Proposition 2 gives the mean of $M_T = s(c_\star) - s(c')$; its variance tells how often the count ranks the pair wrongly for a finite T. By Eq. (1), $M_T = \sum_{t=1}^{T} \xi_t$ with independent terms $\xi_t = \mathbb{1}[x_t \in D]\big(\mathbb{1}[o(x_t) = c_\star(x_t)] - \mathbb{1}[o(x_t) = c'(x_t)]\big) \in \{-1, 0, 1\}$, where $\Pr(\xi_t = 1) = \Pr(D)p_\star$ and $\Pr(\xi_t = -1) = \Pr(D)p_{\text{wrong}}$. Hence

$$\begin{aligned}\operatorname{Var}(M_T) &= T\Big(\mathbb{E}[\xi_t^2] - \mathbb{E}[\xi_t]^2\Big) \\ &= T\Big(\Pr(D)\,(p_\star + p_{\text{wrong}}) - \Pr(D)^2(p_\star - p_{\text{wrong}})^2\Big) \\ &= T\Pr(D)\Big(p_\star + p_{\text{wrong}} - \Pr(D)\,\mu^2\Big),\end{aligned} \tag{8}$$

and by the central limit theorem the probability that the pair is ranked wrongly behaves as

$$\Pr(M_T < 0) \;\approx\; \Phi\left(\frac{\sqrt{T\Pr(D)}\,\mu}{\sqrt{p_\star + p_{\text{wrong}} - \Pr(D)\,\mu^2}}\right), \tag{9}$$

where Φ is the standard normal distribution function. The argument grows as $\sqrt{T}$, so Eq. (9) tends to one when $\mu > 0$ and to zero when $\mu < 0$. For a pair with the pooled L1 values of Table 1, $\Pr(D) = 0.1222$, $p_\star = 0.3718$ and $p_{\text{wrong}} = 0.5105$, the mean of M_T is -1.69 at $T = 100$ and -3.39 at $T = 200$ and the standard deviation is 3.28 and 4.64. The exact probability that the pair is ranked wrongly is 0.64 at $T = 100$ and 0.73 at $T = 200$, with a tie in a further 0.11 and 0.07, and Eq. (9) gives 0.70 and 0.77. More tests make such a pair more likely to be ranked wrongly.

B.4 Proof of Theorem 1

(i) The terms ξ_t of Appendix B.3 are independent and identically distributed with mean $\mathbb{E}[\xi_t] = \Pr(D)(p_\star - p_{\text{wrong}}) = -\Pr(D)\mu$. By the strong law of large numbers, $M_T/T \to -\Pr(D)\mu$ almost surely. If $\mu > 0$ the limit is negative, so $\Pr(M_T < 0) \to 1$ and the count ranks c' above $c_\star$.

(ii) Under Assumption 1, the answer of the oracle at an input $x \in D$ has probability $p_\star$, p_{wrong} or p_{other} of being $c_\star(x)$, $c'(x)$ or another output when $c_\star$ is correct, and p_{wrong}, $p_\star$ or p_{other} when c' is correct. The log-likelihood ratio of one answer is therefore $\log(p_\star/p_{\text{wrong}})$, $\log(p_{\text{wrong}}/p_\star)$ or 0, and it is 0 at every input outside D, where the two hypotheses predict the same output. With $\lambda = \log(p_\star/p_{\text{wrong}})$ this gives

$$L_T = \sum_{t=1}^{T} \lambda\,\xi_t = \lambda\,M_T, \tag{10}$$

so the calibrated score of a pair is the count difference multiplied by λ. Its mean is

$$\mathbb{E}[L_T] = \lambda\,\mathbb{E}[M_T] = T\Pr(D)\,(p_\star - p_{\text{wrong}})\log\frac{p_\star}{p_{\text{wrong}}} = T\Pr(D)\,K(p_\star, p_{\text{wrong}}), \tag{11}$$

which is positive whenever $p_\star \neq p_{\text{wrong}}$, because the two factors of K have the same sign. K is the Kullback–Leibler divergence between $(p_\star, p_{\text{wrong}}, p_{\text{other}})$ and $(p_{\text{wrong}}, p_\star, p_{\text{other}})$, the answer distributions of the two hypotheses. For the tail, Markov's inequality applied to e^{-sL_T} with $s \geq 0$ and the independence of the inputs give

$$\begin{aligned}\Pr(L_T \leq 0) &\leq \mathbb{E}\big[e^{-sL_T}\big] = \Big(\mathbb{E}\big[e^{-s\lambda\xi_t}\big]\Big)^T \\ &= \Big(1 - \Pr(D) + \Pr(D)\big(p_\star e^{-s\lambda} + p_{\text{wrong}}e^{s\lambda} + p_{\text{other}}\big)\Big)^T \\ &= \Big(1 - \Pr(D) + \Pr(D)\big(2\sqrt{p_\star p_{\text{wrong}}} + p_{\text{other}}\big)\Big)^T \qquad \text{at } s = \tfrac{1}{2} \\ &= \Big(1 - \Pr(D)\big(p_\star + p_{\text{wrong}} - 2\sqrt{p_\star p_{\text{wrong}}}\big)\Big)^T \\ &= \Big(1 - \Pr(D)\,\big(\sqrt{p_\star} - \sqrt{p_{\text{wrong}}}\big)^2\Big)^T \;\leq\; e^{-T\Pr(D)\,h},\end{aligned} \tag{12}$$

where the third line uses $e^{\mp\lambda/2} = \sqrt{p_{\text{wrong}}/p_\star}^{\pm 1}$, the fourth uses $p_{\text{other}} = 1 - p_\star - p_{\text{wrong}}$, and the last step uses $1 - u \le e^{-u}$. The choice $s = \frac{1}{2}$ minimizes $p_\star e^{-s\lambda} + p_{\text{wrong}} e^{s\lambda}$, so no other s gives a smaller bound of this form. The bound holds for either sign of λ; when $\mu > 0$, $\lambda < 0$ and Eq. (10) turns the likely negative M_T into a positive L_T, so the calibrated score still ranks $c_\star$ first.

(iii) A reliable answer on $D(c_\star, c')$ adds $\lambda_r \eta_k$ to L_T, where $\lambda_r = \log \frac{1-\varepsilon}{\varepsilon}$ and η_k is $+1$ with probability $1-\varepsilon$ and -1 otherwise, independently of the inputs. Its mean is $(1-2\varepsilon)\lambda_r = K(1-\varepsilon, \varepsilon)$, and the same computation as in Eq. (12) gives the factor

$$\mathbb{E}\big[e^{-\lambda_r \eta_k/2}\big] = (1-\varepsilon)\sqrt{\tfrac{\varepsilon}{1-\varepsilon}} + \varepsilon\sqrt{\tfrac{1-\varepsilon}{\varepsilon}} = 2\sqrt{\varepsilon(1-\varepsilon)} = 1 - \big(\sqrt{1-\varepsilon} - \sqrt{\varepsilon}\big)^2 \tag{13}$$

per answer. Under the vote the same answer adds η_k to M_T, whose expected value is $1 - 2\varepsilon$. □

B.5 From a Pair to the Pool

DisJudge selects a wrong candidate only if some wrong candidate c' has $L_T(c_\star, c') \le 0$ for a correct $c_\star$. If Assumption 1 holds for every such pair, with parameters $\Pr(D_{c'})$ and $h_{c'}$, the union bound and Eq. (12) give

$$\begin{aligned}\Pr(\text{wrong selection}) &\le \sum_{c' \text{ wrong}} \Pr\big(L_T(c_\star, c') \le 0\big) \le \sum_{c' \text{ wrong}} \big(1 - \Pr(D_{c'})\, h_{c'}\big)^T \\ &\le (n-1)\exp\Big(-T \min_{c' \text{ wrong}} \Pr(D_{c'})\, h_{c'}\Big).\end{aligned} \tag{14}$$

The bound needs $\Pr(D_{c'}) > 0$: a wrong candidate that agrees with $c_\star$ on every input the generator produces can only be separated by a tie rule, and such pairs add to the ties of Fig. 1(b). It also needs a correct candidate in the pool, which pass@20 in Table 2 shows is not always there.

B.6 Proof of Proposition 3

Every reliable call is made on an input of $D(c_\star, c')$ and returns $c_\star(x)$ with probability $1-\varepsilon$ and $c'(x)$ with probability ε, independently of the other calls and of the count, as in Proposition 3.

(i) Added as a test, the answer raises $s(c_\star)$ by one in the first case and $s(c')$ by one in the second, so the margin $s(c') - s(c_\star)$ changes by -1 with probability $1-\varepsilon$ and by $+1$ otherwise. After m calls its expectation is $\Delta - m(1-2\varepsilon)$, which is at most zero exactly when $m \ge \Delta/(1-2\varepsilon)$.

(ii) A call contradicts the wrong candidate when the answer is $c_\star(x)$, which has probability $1-\varepsilon$ whatever Δ is, and c' is removed. The correct candidate is contradicted when the answer is $c'(x)$, so it survives m calls with probability $(1-\varepsilon)^m$. At $\varepsilon = 0.08$ and $m = 6$ this is $0.92^6 = 0.606$.

(iii) Under the soft update one call changes $\log w_{c_\star} - \log w_{c'}$ by $+\lambda_r$ with probability $1-\varepsilon$ and by $-\lambda_r$ otherwise, whose mean is $(1-2\varepsilon)\lambda_r = K(1-\varepsilon, \varepsilon)$; no weight is set to zero because every calibrated probability is positive. For the binary submodel, let the k-th call have two possible outputs and error ε_k, so that the reliable source returns the true output with probability $1-\varepsilon_k$. Starting from equal weights,

$$\begin{aligned}\log w_c &= \sum_k \Big(\mathbb{1}[c(x_k) = y_k]\log(1-\varepsilon_k) + \big(1 - \mathbb{1}[c(x_k) = y_k]\big)\log \varepsilon_k\Big) \\ &= \sum_k \log \varepsilon_k + \sum_k \mathbb{1}[c(x_k) = y_k]\, \log \frac{1-\varepsilon_k}{\varepsilon_k} \\ &= \text{const} + \sum_k \lambda_k\, \mathbb{1}[c(x_k) = y_k],\end{aligned} \tag{15}$$

where the constant is shared by all candidates. With a constant $\varepsilon_k = \varepsilon < \frac{1}{2}$ the weight is $\lambda_r > 0$ times the number of agreements, which is the plain vote. With $\varepsilon_k = \frac{1}{2}$ the call adds the same amount to every candidate. As $\varepsilon_k \to 0$ for every k, $\lambda_k \to \infty$, so any candidate with a disagreement falls below every candidate without one, which is the ranking of hard elimination under the same tie rule. □

B.7 The Multi-Valued Update and the Count

Program outputs are rarely binary, and Eq. (3) is stated for a full confusion matrix. For two candidates and one answer y at input x of type q, the update changes their log-odds by

$$\log \frac{w_c}{w_{c'}} \;\mathrel{+}=\; \log \frac{P\big(y \mid c(x), q\big)}{P\big(y \mid c'(x), q\big)}. \tag{16}$$

If $y = c(x) \neq c'(x)$, the numerator is the probability that the source reports the truth and the denominator the probability that it reports $c(x)$ when the truth is $c'(x)$. If y equals neither output, both candidates are contradicted, and Eq. (16) still moves the odds toward the candidate whose output the source confuses with y more often. If the confusion matrix has one accuracy α and spreads its errors evenly over $V-1$ other outputs, every contradicted candidate receives the same factor, and Eq. (3) ranks the candidates by

$$\sum_t \Big(\mathbb{1}[c(x_t) = y_t] \log \alpha + \mathbb{1}[c(x_t) \neq y_t] \log \frac{1-\alpha}{V-1} \Big) = \text{const} + \log \frac{\alpha(V-1)}{1-\alpha} \sum_t \mathbb{1}[c(x_t) = y_t]. \tag{17}$$

For the answers of the oracle, the right-hand side is the count whenever $\alpha(V-1) > 1-\alpha$. The count is therefore the soft update for an oracle that is equally reliable on every input and errs uniformly, an assumption that Section 3.2 finds violated on D. Hard elimination is Eq. (16) with $P(y \mid z, q) = \mathbb{1}[y = z]$, and the vote is Eq. (16) with a factor that depends only on whether $y = c(x)$.

B.8 Estimating the Weight of the Oracle

The soft update weighs an agreement of the oracle on D by $\lambda = \log(p_\star / p_{\text{wrong}})$, and in practice $p_\star$ and p_{wrong} are replaced by the shares $\hat{p}_\star$ and $\hat{p}_{\text{wrong}}$ observed on N calibration inputs. These shares are multinomial, with $\mathrm{Var}(\hat{p}_\star) = p_\star(1-p_\star)/N$, $\mathrm{Var}(\hat{p}_{\text{wrong}}) = p_{\text{wrong}}(1-p_{\text{wrong}})/N$ and $\mathrm{Cov}(\hat{p}_\star, \hat{p}_{\text{wrong}}) = -p_\star p_{\text{wrong}}/N$. The delta method gives

$$\begin{aligned} \mathrm{Var}(\hat{\lambda}) &\approx \frac{\mathrm{Var}(\hat{p}_\star)}{p_\star^2} + \frac{\mathrm{Var}(\hat{p}_{\text{wrong}})}{p_{\text{wrong}}^2} - \frac{2\,\mathrm{Cov}(\hat{p}_\star, \hat{p}_{\text{wrong}})}{p_\star\, p_{\text{wrong}}} \\ &= \frac{1}{N}\left(\frac{1-p_\star}{p_\star} + \frac{1-p_{\text{wrong}}}{p_{\text{wrong}}} + 2 \right) \\ &= \frac{1}{N}\left(\frac{1}{p_\star} + \frac{1}{p_{\text{wrong}}} \right), \end{aligned} \tag{18}$$

so the estimated weight has the wrong sign with probability

$$\Pr\big(\operatorname{sign}\hat{\lambda} \neq \operatorname{sign}\lambda\big) \;\approx\; \Phi\left(-\frac{\big|\log(p_\star/p_{\text{wrong}})\big|\,\sqrt{N}}{\sqrt{1/p_\star + 1/p_{\text{wrong}}}} \right). \tag{19}$$

With the pooled values of Table 1 and $N = 200$, the calibration size of Table 6, this probability is 0.02 at L1, 0.10 at L2, 0.37 at L3 and 0.16 at L4. Near $\mu = 0$ the oracle carries little evidence, as h in Theorem 1 shows, and the sign of its weight is also hard to estimate. Both effects are largest at L3, where DisJudge is least accurate among the model-written oracles (58.3% in Table 1).

C Implementation Details

C.1 Candidate Pools and Oracles

For each of the six benchmarks and each generator we build $4{,}096$ instances. Tasks are reused across instances, and every instance has its own pool of $n = 20$ candidates sampled from the generator and $T = 100$ test inputs written by the generator; mutation-based and property-based inputs replace the model-written ones in Appendix E.6. A candidate is correct if it passes the ground-truth tests of the benchmark. Table 15 lists the statistics of every pool: the pass@20 coverage, $\Pr(D)$, the share of instances with an empty D, μ of the default oracle, and the share of D that the reference covers.

The oracle of level L1 is the generator itself, asked for the expected output of each input with the prompt that produced the candidates. L2 uses the same model with another prompt, L3 a

larger model of the same family, L4 a model of another family, and L5 the bounded-exhaustive reference of Appendix C.2. When the oracle changes, the candidates and the inputs stay fixed. The error agreement rate ρ is measured on HumanEval+, LiveCodeBench and CodeContests as the share of wrong candidate outputs that the oracle reproduces exactly, following Kim et al. (2025).

C.2 Reliable Source and Budget

The reliable source is a bounded-exhaustive reference in the sense of Boyapati et al. (2002): it answers an input only when the input lies within a size bound where exhaustive checking is exact, and it abstains elsewhere. It covers 83.3% of the disagreement inputs of the 32B pools on average, and it cannot reach errors that only appear beyond the bound, such as overflow or excessive running time (Appendix F.5). An abstention leaves every weight unchanged but still costs a call. The budget is set as an expected number of calls per disagreement input, $m/N_D = 0.2$ in the main experiments, so the number of calls grows with N_D, and an instance never makes more calls than N_D. For the exact reference, whose error is zero, the soft update uses a small positive floor in place of this error, small enough that one reliable answer outweighs all answers of the oracle on D. Every rule that queries the reliable source receives the same budget, which is 1.92 calls per instance in the default setting.

C.3 Calibration of the Confusion Matrices

The confusion matrix of each oracle is calibrated on labeled disagreement inputs. Table 6 reports the calibrated error of each oracle on D, with the 95% interval of 200 calibration inputs, together with the precision, recall and cost of three checks of the implementation. For a pair, the soft update weighs an agreement with the oracle on D by $\log(p_\star/p_{\text{wrong}})$; this weight is negative at L1 to L3, where $\mu > 0$, and positive at L4, and Appendix B.8 shows how often its sign is estimated wrongly.

Table 6: Implementation checks and calibration of the sources. Precision and recall of three checks over all inputs, in percent, with their cost per check. Calibrated error of each source on D, in percent, with the 95% interval from 200 calibration inputs, and the cost of one call; the reference is exact where it answers.

Implementation check	Precision	Recall	Cost (ms)
Membership in D agrees with the pairwise test	100.00	100.00	0.37
Expected output parses	99.73	99.42	0.44
An answer outside D changes no margin	100.00	100.00	1.26
Source	**Error on D [95% interval]**		**Cost (ms)**
L1, same model, same prompt	62.82 [56.12, 69.52]		7.88
L2, same model, other prompt	59.67 [52.87, 66.47]		5.90
L3, larger model, same family	58.22 [51.39, 65.06]		6.42
L4, other model family	56.70 [49.83, 63.57]		8.01
L5, executable reference	0.00 (exact)		6.54

In the controlled simulation the error of the source is estimated per query type from 16, 64 or 128 calibration records. Table 7 and Fig. 3 report the bias and the RMSE of this estimate over 3,072 query records per benchmark. From 16 to 128 records, the median bias falls by a factor of 7.5 to 8.5 and the median RMSE by a factor of 2.7 to 3.4. These are the rates $1/N$ and $1/\sqrt{N}$ expected for the bias of a smoothed frequency and for its sampling error, so more records remove the bias quickly while the sampling error falls slowly. Table 26 shows what these errors cost in selection accuracy.

C.4 The Algorithm

Algorithm 1 states the full procedure for one instance. The vote and hard elimination differ from it only in the update line. Recomputing g after every answer gives the adaptive order of Table 27.

C.5 Controlled Simulation

The simulation isolates how the answers of a noisy source are used. For each of the six benchmarks it contains 256 instances. An instance is a binary matrix of four candidates by twelve inputs, where an entry says whether the output of the candidate equals the true output; one candidate is correct

Table 7: Bias and RMSE of the estimated error rate in the controlled simulation, by true error rate ε and number of calibration records (16, 64 or 128); mean, minimum and maximum over the six benchmarks.

		Bias			RMSE		
ε	Records	Mean	Min	Max	Mean	Min	Max
0.02	16	0.0285	0.0279	0.0289	0.0444	0.0437	0.0455
0.02	64	0.0074	0.0071	0.0079	0.0193	0.0189	0.0202
0.02	128	0.0037	0.0036	0.0039	0.0132	0.0130	0.0134
0.08	16	0.0248	0.0234	0.0259	0.0693	0.0685	0.0704
0.08	64	0.0065	0.0060	0.0069	0.0349	0.0344	0.0360
0.08	128	0.0033	0.0029	0.0036	0.0246	0.0245	0.0248
0.2	16	0.0176	0.0161	0.0213	0.0954	0.0937	0.0973
0.2	64	0.0044	0.0035	0.0053	0.0495	0.0488	0.0508
0.2	128	0.0020	0.0016	0.0025	0.0353	0.0347	0.0357

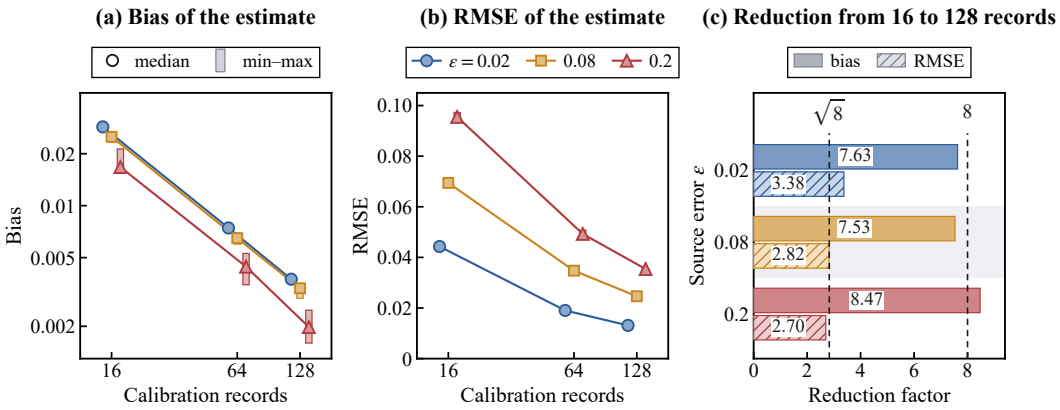

Figure 3: Estimating the error rate of a source (controlled simulation). (a) Bias and (b) RMSE of the estimate against the number of records, one line per error rate; markers are medians and bars span the six benchmarks. (c) Factor by which bias and RMSE shrink from 16 to 128 records, with reference values 8 and $\sqrt{8}$.

on every input unless a row of Table 4 says otherwise, and D holds the inputs on which the four entries are not all equal. The source returns the true answer or a wrong one, with an error that varies across queries around a mean of $\bar{\varepsilon} = 0.08$, and the update uses error rates estimated from calibration records per query type, whose number Table 26 varies. Six calls are requested per instance and 5.99 are made on average, since an instance never makes more calls than N_D. Without any answer the selection is at chance (24.0%). The simulated sources S1 to S5 of Appendix F.4 differ in their error on D: S1 is the default source, S2 and S3 make more errors on D than S1, S4 has errors correlated with those of the candidates, and S5 is an exact reference that covers and answers 80% of the queries.

C.6 A Worked Example

We follow one instance through Algorithm 1 to show each branch. Four candidates run on twelve inputs, and Table 8 shows which outputs are right. Candidate c_1 is correct. On each input the wrong candidates return the same wrong output, so every input of $D = \{x_2, x_4, x_5, x_7, x_9, x_{11}\}$ splits the candidates into two groups. The oracle is wrong on x_4, where it returns the output of c_3, and on x_7, where it returns the output of c_2 and c_4. The counts are then 10, 10, 9 and 8, so the correct c_1 is tied with the wrong c_2. We set the calibrated oracle to $p_\star = 2/3$ and $p_{\text{wrong}} = 1/3$, its accuracy on this D, and the reliable source to $\varepsilon = 0.08$, so an agreement with a reliable answer is worth $\lambda_r = \log(0.92/0.08) = 2.44$. Three calls are allowed, and ties in g are broken by input order. The vote uses the same three answers, and hard elimination uses them until one candidate remains.

Initial weights. On D, candidate c agrees with the oracle on 4, 4, 3 and 2 inputs. With $p_\star = 2/3$ and $p_{\text{wrong}} = 1/3$ each agreement is worth $\log 2 = 0.69$, so the relative log-weights are 0, 0, -0.69 and -1.39. They follow the count, since this oracle has $\mu < 0$, and c_1 and c_2 are still tied.

First call. The weights are 1, 1, 0.5 and 0.25, and g is 0.55 at x_2 and x_7, 0.73 at x_{11}, 0.82 at x_4 and x_5, and 0.91 at x_9. The first of the two best inputs, x_2, is queried, and the source returns

Algorithm 1 DisJudge

Input: candidates $c_1, \ldots, c_n$; T test inputs x with expected outputs $o(x)$ and query types q_x; calibrated confusion matrices P_o and P_r; budget m.
Execute every candidate on every input; let D be the inputs on which at least two candidates differ, and $N_D = |D|$.
$\log w_c \leftarrow \sum_{x \in D} \log P_o\big(o(x) \mid c(x), q_x\big)$ for every candidate c.
for $k = 1, \ldots, \min(m, N_D)$ **do**
    For every unqueried $x \in D$, compute $g(x) = \max_v \sum_{c:\, c(x)=v} w_c / \sum_c w_c$.
    Query the reliable source at $x_k = \arg\min_x g(x)$, ties broken by input order, and receive y_k; if it abstains, go to the next iteration.
    $\log w_c \mathrel{+}= \log P_r\big(y_k \mid c(x_k), q_{x_k}\big)$ for every candidate (soft update)
end for
Output: $\arg\max_c w_c$, ties broken by the count $s(c)$.

Table 8: The instance of Appendix C.6. An entry 1 means that the candidate returns the true output. Inputs marked D form the disagreement set, and the split gives the number of right and wrong candidates on each.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
c_1 (correct)	1	1	1	1	1	1	1	1	1	1	1	1
c_2	1	0	1	1	1	1	0	1	1	1	1	1
c_3	1	1	1	0	0	1	1	1	1	1	0	1
c_4	1	0	1	1	1	1	0	1	0	1	0	1
In D		D		D	D		D		D		D	
Split		2:2		3:1	3:1		2:2		3:1		2:2	

the true output, which agrees with c_1 and c_3. The log-weights become 2.44, 0, 1.75 and -1.39, so c_1 leads. The vote gives the counts 11, 10, 10 and 8, and hard elimination removes c_2 and c_4.

Second call, with a wrong answer. Now g is lowest at x_{11} (0.68), where c_1 and c_2 face c_3 and c_4. The source errs and returns the output of c_3 and c_4. The log-weights become 2.44, 0, 4.19 and 1.06, so the wrong c_3 leads, but no candidate is removed. The vote ties c_1 and c_3 at 11. Hard elimination removes the correct c_1 permanently; only c_3 survives, so it stops querying and returns c_3.

Third call. The soft update next asks x_4, where $g = 0.81$ and c_3 stands alone. The correct answer contradicts c_3. The log-weights become 4.88, 2.44, 4.19 and 3.50, so c_1 is selected. It leads c_3 by 0.69: both agree with two of the three reliable answers, and the calibrated oracle breaks the tie. The vote also ends with c_1, at counts 12, 11, 11 and 10, while hard elimination already returned c_3.

Summary. In this instance, one wrong answer out of three makes hard elimination fail permanently, the risk stated in Proposition 3(ii), while the soft update keeps every candidate and recovers. In this instance the oracle has $\mu < 0$ and only breaks the final tie; when $\mu > 0$ its calibrated weight is negative and it moves the ranking against the candidates it agrees with, which is what separates the soft update from the vote on the full pools, where 31.0% of the candidate pairs have $\mu > 0$.

C.7 Other Selection Rules

All rules run on the same pools and the same test inputs. The count keeps the candidate that matches the most expected outputs. CodeT scores candidates by dual agreement with generated tests (Chen et al., 2023). AlphaCode clustering groups candidates by their outputs and picks from the largest cluster (Li et al., 2022). MBR-Exec minimizes an execution-based Bayes risk (Shi et al., 2022), and execution self-consistency keeps the most common behavior (Wang et al., 2023). LEVER, Coder-Reviewer reranking, outcome and process reward models and the fault-aware ranker score candidates with trained or prompted models (Ni et al., 2023; Zhang et al., 2023; Cobbe et al., 2021; Lightman et al., 2024; Inala et al., 2022). The pairwise LLM judge and the tournament compare candidates in pairs, and self-critique scores each candidate with its own critique (Zheng et al., 2023; Saunders et al., 2022). The test-designer agent writes additional tests (Huang et al., 2023), Self-Debugging revises candidates with execution feedback (Chen et al., 2024), and S* combines sampling with adaptive input synthesis for selection (Li et al., 2025). The query rules ask the reliable source at distinguishing inputs (Jha et al., 2010), at inputs of largest committee

disagreement (Seung et al., 1992), or by noisy generalized binary search (Nowak, 2009). The three query rules just above DisJudge in Table 2 spend the same budget inside D: in the order of the score margin between candidates, only on inputs where the two leading candidates differ, or in random order. Rules marked ✓ in Table 2 query the reliable source at the same budget as DisJudge.

D Additional Diagnosis Results

D.1 Number of Tests and the Margin

Fig. 4 follows the count from 10 to 200 tests on the 32B pools. The average pass rate of the candidates falls by 0.5 to 0.9 points, and the selection accuracy of the count rises by 0.3 to 0.4 points (panel a). At $T = 200$ the count stays 15 to 22 points below the pass@20 upper bound, although the oracle is right on 84% to 90% of all inputs (panel b). Panel (c) compares the quantiles of the observed margin between the correct and the selected candidate with the prediction of Eq. (2), on HE+, LCB, BCB and CC at 50 quantiles each. The largest gap is 0.096 tests and the mean gap 0.028.

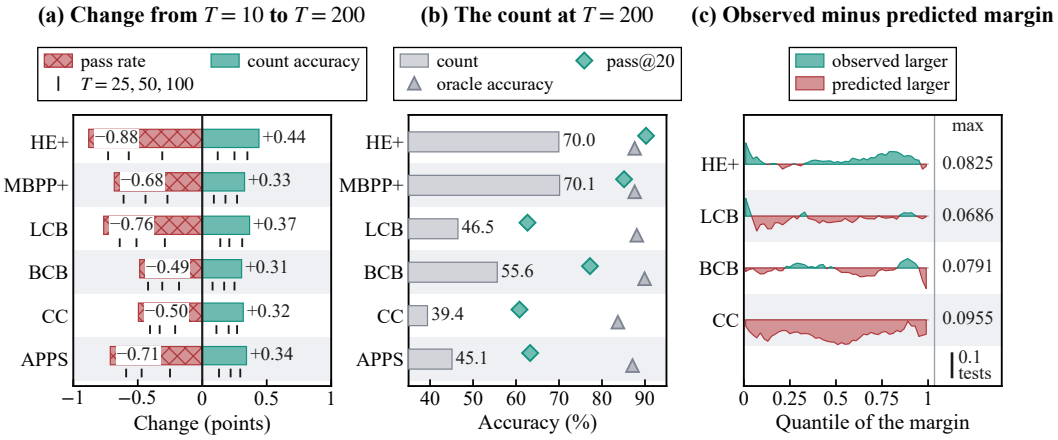

Figure 4: The count as the number of tests grows (32B pools). (a) Change of the average pass rate and of the selection accuracy of the count from $T = 10$ to $T = 200$; ticks mark $T = 25$, 50 and 100. (b) Accuracy of the count at $T = 200$, with the pass@20 bound and the overall accuracy a of the oracle. (c) Observed minus predicted margin across 50 quantiles on HE+, LCB, BCB and CC; the right column gives the largest gap.

Table 9: Values behind Fig. 4(a) and (b) (32B pools, per benchmark). Cumulative change, in points, of the average pass rate and of the selection accuracy of the count from $T = 10$ to $T = 25$, 50, 100 and 200; at $T = 200$, the overall accuracy a of the oracle, the accuracy of the count and the pass@20 upper bound (%).

	Change of the pass rate				Change of the count's accuracy				At $T = 200$		
	25	50	100	200	25	50	100	200	a	Count	pass@20
HE+	−0.310	−0.570	−0.730	−0.880	+0.120	+0.250	+0.350	+0.440	87.6	69.96	90.25
MBPP+	−0.270	−0.440	−0.610	−0.682	+0.090	+0.180	+0.270	+0.328	87.6	70.14	85.13
LCB	−0.290	−0.510	−0.640	−0.764	+0.140	+0.210	+0.310	+0.367	88.1	46.49	62.67
BCB	−0.180	−0.310	−0.420	−0.485	+0.080	+0.170	+0.250	+0.305	89.9	55.63	77.20
CC	−0.210	−0.330	−0.405	−0.495	+0.110	+0.205	+0.270	+0.318	83.7	39.40	60.80
APPS	−0.250	−0.470	−0.590	−0.713	+0.130	+0.220	+0.295	+0.343	87.1	45.11	63.28

D.2 Error Breakdown per Benchmark

The values behind Fig. 1(b) and (c) are listed per benchmark in Table 10, together with a further measurement: a pair of oracles, A and B, whose overall accuracies are matched on each benchmark but whose accuracies on D differ. Their selection accuracies differ by 7.8 to 8.9 points across the six benchmarks, with a 95% interval of about ±1.3 points. Bias on D and random errors or ties are separated by the sign of μ for the selected and a correct candidate, as in Section 3.2 and Fig. 1(b).

Table 10: Per-benchmark values behind Fig. 1 (32B pools, %). All wrong selections split into bias on D (Bias), pools with no correct candidate (None) and random errors or ties (Rand.); wrong selections in pools that hold a correct candidate $c_\star$ and flip under the reference (Flip), split into bias and random flips; oracles A and B with matched overall accuracy a, their selection accuracies (Sel. A, B), the difference and the half-width ($\pm$) of its 95% interval; shares of inputs off and in D; share of D the reference covers (Cov.).

	All wrong selections			Pool holds $c_\star$			Matched overall accuracy					Inputs		
	Bias	None	Rand.	Flip	Bias	Rand.	a	Sel. A	Sel. B	Diff.	$\pm$	off D	in D	Cov.
HE+	51.40	32.00	16.61	41.23	33.29	7.94	87.83	72.20	63.66	8.54	1.13	90.38	9.62	86.68
MBPP+	36.57	49.25	14.18	42.38	33.45	8.93	88.13	73.03	64.97	8.06	1.37	86.58	13.42	87.62
LCB	23.28	69.30	7.43	42.26	27.06	15.21	87.22	49.65	41.10	8.55	1.35	90.12	9.88	85.70
BCB	30.35	51.04	18.61	39.88	25.40	14.47	87.90	58.04	50.27	7.77	1.22	86.59	13.41	80.52
CC	27.55	64.35	8.10	39.98	29.12	10.86	87.48	42.64	33.77	8.87	1.24	84.29	15.71	78.52
APPS	21.29	66.49	12.23	40.45	26.18	14.27	84.60	48.23	39.40	8.83	1.31	88.75	11.25	80.61
Mean	31.74	55.40	12.86	41.03	29.08	11.95	87.19	57.30	48.86	8.44	1.27	87.79	12.22	83.27

D.3 Overall Accuracy and the Accuracy Outside D

Solving Eq. (7) for the accuracy outside D gives $a_{\text{off}} = (a - \Pr(D)\, p_\star)/(1 - \Pr(D))$. Table 11 evaluates it at the six-benchmark means of Table 1 with $\Pr(D) = 0.1222$. At the model-written levels the oracle is right on 94% to 96% of the inputs outside D and on 37% to 43% of the inputs in D. Because D holds only an eighth of the inputs, it contributes 4.5 to 5.3 points to the overall accuracy, and a change of 0.06 in $p_\star$ moves a by less than one point. L2 has the highest overall accuracy because it is the most accurate outside D, on inputs that never affect the ranking.

Table 11: The overall accuracy decomposed (six-benchmark means, %): accuracy on D, overall accuracy, implied accuracy outside D at $\Pr(D) = 0.1222$, and the contribution $\Pr(D)\, p_\star$ of D to the overall accuracy.

	Oracle	$p_\star$ on D	a	a_{off}	$\Pr(D)\, p_\star$
L1	Same model, same prompt	37.18	87.33	94.31	4.54
L2	Same model, other prompt	40.33	89.02	95.80	4.93
L3	Larger model, same family	41.78	88.67	95.20	5.11
L4	Other model family	43.30	87.83	94.03	5.29
L5	Executable reference	100.00	100.00	100.00	12.22

E Additional Results on the Candidate Pools

E.1 All Rules in the Default Setting

We repeat every rule of the default setting on three independent candidate pools (Table 12). The bootstrap interval of DisJudge is $[74.72, 76.22]$. In this setting the difference intervals of AlphaCode clustering and Self-Debugging contain zero, and all other differences from DisJudge exclude it.

E.2 Safety and Cost

For every rule that queries the reliable source, Table 13 reports four safety rates in the default setting, each with its own denominator. The soft update and the vote remove nothing. Hard elimination removes 62.3% of the wrong candidates and keeps every correct one, since the reference is exact where it answers; the 0.19% of pools it empties hold no correct candidate. The four rules that eliminate by their own criteria keep 97.3% to 98.0% of the correct candidates. All rules pay the same 1.92 calls per instance, 11.2 ms per calling instance; 1.93% of the instances have an empty D, the soft update is right on 66.5% of them, and the whole pipeline takes 138.6 ms per instance.

E.3 Gains in All Settings

DisJudge gains over the count in all 18 settings of Table 2 (Table 14), by 4.82 to 6.54 points on the pool of Table 2 and by 4.79 to 6.48 points on average over three further independent pools.

Table 12: All rules in the default setting, HumanEval+ with the 32B pool (%): the point estimate of Table 2, the accuracy on three independent pools with their mean and standard deviation, the 95% bootstrap interval of the mean (2,000 resamples), and the difference to DisJudge with its interval; grey rows are reference points.

		Independent pools					95% interval		Difference to soft update		
Rule	Point	1	2	3	Mean	SD	Lower	Upper	Diff.	Lower	Upper
Random choice	37.29	37.24	37.17	37.40	37.27	0.12	36.58	37.99	−38.23	−38.93	−37.42
Count of passed tests	69.87	70.03	69.97	69.85	69.95	0.09	69.11	70.63	−5.55	−6.36	−4.67
Upper bound (pass@20)	90.25	90.35	90.30	89.98	90.21	0.20	89.59	90.97	14.71	13.83	15.35
CodeT	71.50	71.54	71.40	71.65	71.53	0.13	70.90	72.32	−3.97	−4.71	−3.25
AlphaCode clustering	75.50	75.58	75.50	75.57	75.55	0.04	74.97	76.19	0.05	−0.62	0.90
MBR-Exec	72.40	72.21	72.29	72.58	72.36	0.19	71.73	73.21	−3.14	−3.83	−2.40
Execution self-consistency	73.20	73.09	73.17	73.22	73.16	0.07	72.39	73.75	−2.34	−3.14	−1.49
LEVER	69.80	69.94	69.84	69.62	69.80	0.16	69.17	70.43	−5.70	−6.52	−4.82
Coder-Reviewer reranking	73.90	73.99	73.83	73.97	73.93	0.09	73.16	74.66	−1.57	−2.41	−0.79
Outcome/process reward model	71.70	71.75	71.79	71.80	71.78	0.03	71.24	72.54	−3.72	−4.53	−2.99
Fault-aware code ranker	69.20	69.20	69.36	69.16	69.24	0.11	68.54	69.89	−6.26	−7.07	−5.59
LLM judge, pairwise	71.40	71.45	71.42	71.42	71.43	0.02	70.82	72.16	−4.07	−4.90	−3.34
Tournament elimination	71.10	71.18	71.16	71.23	71.19	0.04	70.62	71.86	−4.31	−4.95	−3.44
Self-critique scoring	73.40	73.28	73.49	73.49	73.42	0.12	72.87	74.23	−2.08	−2.75	−1.33
More generated tests	74.20	74.12	74.37	74.26	74.25	0.13	73.47	75.03	−1.25	−1.88	−0.45
Test-designer agent	72.70	72.63	72.64	72.86	72.71	0.13	71.98	73.35	−2.79	−3.41	−2.10
Self-Debugging	75.00	74.85	74.98	74.99	74.94	0.08	74.12	75.71	−0.56	−1.33	0.30
S^*	70.50	70.53	70.59	70.56	70.56	0.03	69.84	71.37	−4.94	−5.75	−4.13
Distinguishing inputs	71.40	71.40	71.29	71.33	71.34	0.06	70.56	72.13	−4.16	−4.99	−3.27
Query by committee	72.70	72.53	72.61	72.78	72.64	0.13	72.01	73.43	−2.86	−3.66	−2.24
Noisy GBS	71.90	71.99	71.94	71.89	71.94	0.05	71.40	72.64	−3.56	−4.35	−2.90
Margin-ordered queries	73.50	73.65	73.62	73.50	73.59	0.08	72.98	74.41	−1.91	−2.75	−1.04
Top-2 queries	69.80	69.88	69.91	69.79	69.86	0.06	69.20	70.44	−5.64	−6.38	−4.96
Random queries within D	73.20	73.25	73.09	73.14	73.16	0.08	72.59	74.01	−2.34	−3.10	−1.57
DisJudge, vote update	73.19	73.18	73.08	73.04	73.10	0.07	72.26	73.92	−2.40	−3.25	−1.61
DisJudge, soft update	75.41	75.50	75.45	75.55	75.50	0.05	74.72	76.22	0.00	0.00	0.00
DisJudge, hard elimination	73.45	73.57	73.40	73.38	73.45	0.10	72.66	74.03	−2.05	−2.77	−1.32

Table 13: Safety and cost in the default setting (%, except calls and cost). Kept: correct candidates kept over all correct candidates. Wrong removed: wrong candidates removed over all wrong ones. Emptied: pools emptied over all pools. Precision: removals that hit a wrong candidate over all removals, undefined when nothing is removed. Accuracy on instances with $N_D > 0$ and overall; expected calls per instance; cost per calling instance.

Rule	Kept	Wrong removed	Emptied	Precision	Acc. $N_D>0$	Acc.	Calls	Cost (ms)
Count of passed tests	100.00	0.00	0.00	–	69.94	69.87	0.00	0.00
LLM judge, pairwise	100.00	0.00	0.00	–	71.50	71.40	1.92	11.19
Tournament elimination	97.31	58.95	0.12	97.35	71.19	71.10	1.92	11.19
More generated tests	100.00	0.00	0.00	–	74.35	74.20	1.92	11.19
Test-designer agent	100.00	0.00	0.00	–	72.82	72.70	1.92	11.19
Distinguishing inputs	97.96	74.79	0.12	98.40	71.50	71.40	1.92	11.19
Query by committee	97.79	73.24	0.05	98.23	72.82	72.70	1.92	11.19
Noisy GBS	98.04	55.48	0.22	97.95	72.01	71.90	1.92	11.19
Margin-ordered queries	100.00	0.00	0.00	–	73.64	73.50	1.92	11.19
Top-2 queries	100.00	0.00	0.00	–	69.87	69.80	1.92	11.19
Random queries within D	100.00	0.00	0.00	–	73.33	73.20	1.92	11.19
DisJudge, vote update	100.00	0.00	0.00	–	73.31	73.19	1.92	11.19
DisJudge, soft update	100.00	0.00	0.00	–	75.58	75.41	1.92	11.19
DisJudge, hard elimination	100.00	62.32	0.19	100.00	73.58	73.45	1.92	11.19

E.4 Statistics of the Pools

The 18 pools differ in difficulty and in the size of D (Table 15). The pass@20 coverage ranges from 57.6% to 98.8%, D holds 9.6% to 17.1% of the inputs, fewer than 2% of instances have an empty D, and the reference covers 73.6% to 87.6% of D. On the instances with an empty D the soft update falls back to the count, and over all instances it spends 1.9 to 3.4 calls per instance, a fifth of N_D on average.

E.5 Budget

We sweep the budget from 0.02 to 1 call per disagreement input on the 32B pools (Fig. 5) for the soft update, the vote, hard elimination and the two query rules that differ from DisJudge only in the order of the calls. On HumanEval+ the soft update leads the other four rules at every budget, and its lead grows from 0.34 points at the smallest budget to 2.59 points when all of D is queried; at the budgets 0.2 and 1 it leads by 1.7 to 3.8 points across benchmarks. Table 17 gives the cost of each budget. The main budget of a fifth of D reaches 75.4% in the default setting at a fifth of the cost of querying

Table 14: Gain of DisJudge over the count (%) on the pool of Table 2 and over three further pools.

				Gain	
Benchmark	Generator	Count	DisJudge	Main pool	Three pools
HE+	7B	71.78	77.40	+5.62	+5.71
HE+	32B	69.87	75.41	+5.54	+5.55
HE+	480B	76.66	82.37	+5.71	+5.58
MBPP+	7B	64.30	70.22	+5.92	+5.86
MBPP+	32B	70.08	75.61	+5.53	+5.67
MBPP+	480B	72.56	78.28	+5.72	+5.86
LCB	7B	45.13	51.09	+5.96	+6.04
LCB	32B	46.43	52.37	+5.94	+5.87
LCB	480B	50.30	55.34	+5.04	+5.17
BCB	7B	48.68	55.22	+6.54	+6.45
BCB	32B	55.57	61.18	+5.61	+5.52
BCB	480B	56.20	61.33	+5.13	+5.16
CC	7B	39.51	45.61	+6.10	+6.22
CC	32B	39.35	44.17	+4.82	+4.79
CC	480B	42.88	49.13	+6.25	+6.16
APPS	7B	40.72	46.62	+5.90	+5.79
APPS	32B	45.07	51.48	+6.41	+6.48
APPS	480B	50.41	55.34	+4.93	+5.00

Table 15: Statistics of the 18 pools. Pass@20 (Cov.), $\Pr(D)$ and $\Pr(N_D = 0)$ in percent; μ of the L1 oracle; coverage of D by the reference (Ref., %); accuracy of the count and the soft update, and of the latter on instances with $N_D > 0$ and $N_D = 0$ (%); calls per instance; cost per calling instance and wall clock, in ms.

		Cov.	$\Pr(D)$	$\Pr(N_D{=}0)$	μ	Ref.	Count	Soft	Acc. $N_D{>}0$	Acc. $N_D{=}0$	Calls	Cost	Wall
HE+	7B	87.73	9.84	1.76	0.132	84.47	71.78	77.40	77.55	69.25	1.97	9.8	230
HE+	32B	90.25	9.62	1.93	0.131	86.68	69.87	75.41	75.58	66.46	1.92	11.2	139
HE+	480B	98.77	12.87	1.80	0.154	78.50	76.66	82.37	82.51	74.47	2.57	14.4	204
MBPP+	7B	85.60	14.77	1.39	0.156	75.78	64.30	70.22	70.35	61.32	2.95	11.0	222
MBPP+	32B	85.13	13.42	1.87	0.128	87.62	70.08	75.61	75.78	66.47	2.68	17.3	221
MBPP+	480B	92.54	14.14	1.57	0.139	74.96	72.56	78.28	78.41	70.02	2.83	10.4	169
LCB	7B	58.97	16.22	1.10	0.155	73.62	45.13	51.09	51.18	42.87	3.24	23.6	185
LCB	32B	62.67	9.88	1.28	0.148	85.70	46.43	52.37	52.49	42.94	1.98	10.2	209
LCB	480B	70.53	11.35	1.24	0.128	78.51	50.30	55.34	55.44	47.51	2.27	15.6	208
BCB	7B	64.73	12.34	0.71	0.155	73.93	48.68	55.22	55.29	45.52	2.47	9.8	171
BCB	32B	77.20	13.41	1.27	0.149	80.52	55.57	61.18	61.28	53.56	2.68	18.9	201
BCB	480B	72.66	14.45	0.94	0.136	78.42	56.20	61.33	61.41	52.45	2.89	20.5	215
CC	7B	57.82	13.61	1.75	0.161	85.93	39.51	45.61	45.77	36.88	2.72	12.8	251
CC	32B	60.80	15.71	1.42	0.124	78.52	39.35	44.17	44.29	35.97	3.14	15.5	206
CC	480B	63.12	17.11	1.87	0.161	80.23	42.88	49.13	49.29	40.92	3.42	19.9	244
APPS	7B	57.57	14.64	0.86	0.144	86.80	40.72	46.62	46.70	37.68	2.93	14.7	214
APPS	32B	63.28	11.25	1.94	0.151	80.61	45.07	51.48	51.68	41.40	2.25	9.6	200
APPS	480B	70.47	13.10	1.15	0.127	74.91	50.41	55.34	55.43	48.03	2.62	17.8	248

all of D, which reaches 77.3%. Table 18 lists the accuracies at two budgets and the share of the gain of the soft update that each other rule captures, and Table 16 gives the values behind Fig. 5(a).

Table 16: Budget sweep on HumanEval+ (32B pool, %): selection accuracy of the five rules of Fig. 5(a) at the six budgets from 0.02 to 1, and the lead in points of the soft update over the best of the other four.

m/N_D	Vote	Soft	Hard	Margin	Rand. D	Soft − best other
0.02	70.16	70.54	70.20	70.06	70.16	+0.34
0.05	70.96	71.83	71.06	70.75	70.96	+0.77
0.1	71.97	73.39	72.15	71.61	71.98	+1.24
0.2	73.19	75.41	73.45	73.50	73.20	+1.91
0.5	74.24	77.02	74.58	73.54	74.26	+2.44
1	74.38	77.33	74.73	73.67	74.41	+2.59

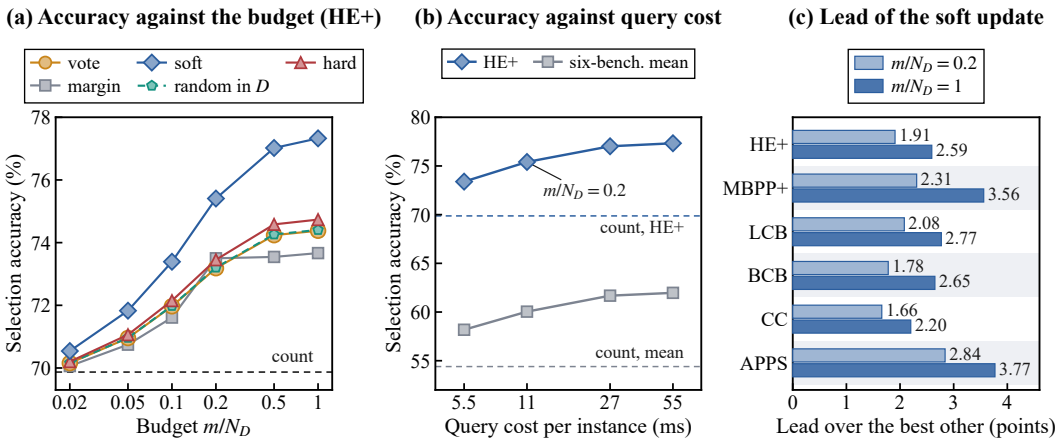

Figure 5: Budget sweep (32B pools). (a) Accuracy on HumanEval+ against the budget m/N_D for the three update rules and the margin and random orders within D; black dashes: the count. (b) Accuracy of the soft update against the query cost per instance on HE+ and over the six benchmarks, at the budgets of Table 17; dashed lines: the count. (c) Lead of the soft update over the best other rule at $m/N_D = 0.2$ and 1, per benchmark.

Table 17: Cost of the budget (32B pools): calls per instance and cost in ms on HumanEval+, and selection accuracy there and over the six benchmarks (%). The shaded row is the main budget, a fifth of N_D.

Budget	Calls per instance	Cost (ms)	HumanEval+	Six-benchmark mean
All of D	9.62	54.88	77.33	61.97
$m = N_D/2$	4.81	27.44	77.02	61.68
$m = N_D/5$ (main)	1.92	10.98	75.41	60.04
$m = N_D/10$	0.96	5.49	73.39	58.18

E.6 Generality

We also run DisJudge along five further axes on four benchmarks (Table 19). Over eight generators, including Llama-3.3-70B (Grattafiori et al., 2024), two Devstral models (Mistral AI, 2025), QwQ-32B (Qwen Team, 2025b) and Qwen3-Coder-30B-A3B (Qwen Team, 2025a), the four-benchmark mean ranges from 57.3% to 62.0%. Mutation-based and property-based inputs give 59.3% and 59.8%, against 58.3% for model-written inputs. Pools of 5 to 50 candidates and four task families stay in the same range. With S* as the first stage, adding DisJudge raises the mean from 54.2% to 60.2%.

E.7 Composition with First-Stage Rules

A first-stage rule selects first and passes its top candidates on, and DisJudge queries the reliable source on their disagreement set with the same budget. In Table 20 the combination improves all four first-stage rules, by 3.4 points for AlphaCode clustering and by 4.3 to 5.9 points for the other three. The increments are close to additive for CodeT and S* and subadditive for clustering and self-consistency.

E.8 Similarity Levels per Benchmark

Per benchmark, the similarity levels behave as in Table 1 (Table 21). On all six benchmarks the count improves from L1 to L5, and DisJudge is most accurate at L1 and least accurate at L3 among the model-written oracles. On four benchmarks the order of DisJudge over L1 to L4 is the order of h; on MBPP+ and CodeContests, L2 and L4 swap places, with accuracies within 0.3 points. Fig. 6 places the other rules along the same levels. The lead of the soft update over the count falls from 5.6 to 0.2 points, and over the strongest other rule from 2.1 to 0.1 points, as the oracle leaves less bias to correct.

Table 18: Two budgets in detail (32B pools, %): selection accuracy of the five rules of Table 16 and the count at $m/N_D = 0.2$ and 1, and the share of the gain of the soft update over the count that each of the other rules reaches.

		Selection accuracy (%)						Share of the soft-update gain (%)			
	m/N_D	Vote	Soft	Hard	Margin	Rand. D	Count	Vote	Hard	Margin	Rand. D
HE+	0.2	73.19	75.41	73.45	73.50	73.20	69.87	59.9	64.6	65.5	60.1
HE+	1	74.38	77.33	74.73	73.67	74.41	69.87	60.5	65.1	50.9	60.9
MBPP+	0.2	72.93	75.61	72.93	70.70	73.30	70.08	51.5	51.5	11.2	58.2
MBPP+	1	73.95	77.51	73.95	73.32	73.60	70.08	52.1	52.1	43.6	47.4
LCB	0.2	49.92	52.37	50.29	47.40	49.10	46.43	58.8	65.0	16.3	44.9
LCB	1	51.16	54.41	51.64	49.52	51.57	46.43	59.3	65.3	38.7	64.4
BCB	0.2	59.01	61.18	58.91	59.30	59.40	55.57	61.3	59.5	66.5	68.3
BCB	1	60.22	63.09	60.08	58.59	60.44	55.57	61.8	60.0	40.2	64.8
CC	0.2	41.93	44.17	42.51	41.40	40.10	39.35	53.5	65.6	42.5	15.6
CC	1	42.86	45.83	43.63	42.32	42.00	39.35	54.2	66.0	45.8	40.9
APPS	0.2	48.13	51.48	48.64	47.50	46.30	45.07	47.7	55.7	37.9	19.2
APPS	1	49.22	53.67	49.90	48.91	47.44	45.07	48.3	56.2	44.7	27.6

Table 19: DisJudge along five axes (selection accuracy, %; Avg: mean over the four benchmarks): pool generator, input generator, pool size n, a first-stage rule placed in front, and the task family. The rows with the 32B pool, model-written inputs and $n = 20$ coincide and repeat the 32B values of DisJudge in Table 2.

Configuration	HE+	LCB	BCB	CC	Avg
Candidate pool					
Qwen2.5-Coder-7B	77.40	51.09	55.22	45.61	57.33
Qwen2.5-Coder-32B	75.41	52.37	61.18	44.17	58.28
Qwen3-Coder-30B-A3B	79.81	50.73	61.46	44.14	59.03
Qwen3-Coder-480B-A35B	82.37	55.34	61.33	49.13	62.04
Llama-3.3-70B	82.10	49.66	61.43	44.63	59.45
Devstral-Small-2-24B	81.03	53.97	63.43	44.00	60.61
Devstral-2-123B	77.74	51.65	62.70	44.93	59.26
QwQ-32B	80.57	52.98	57.38	44.55	58.87
Input generator					
Model-generated inputs	75.41	52.37	61.18	44.17	58.28
Mutation	80.66	50.74	62.49	43.37	59.31
Property-based	75.80	54.79	64.51	43.89	59.75

Configuration	HE+	LCB	BCB	CC	Avg
Pool size					
$n = 5$	82.65	56.40	61.95	43.06	61.02
$n = 10$	77.96	56.37	61.05	49.76	61.29
$n = 20$	75.41	52.37	61.18	44.17	58.28
$n = 50$	79.18	52.36	57.60	44.72	58.46
First stage					
S^* alone	70.50	47.00	56.30	42.80	54.15
S^* + DisJudge	76.39	53.58	63.65	47.22	60.21
Task family					
Function-level	74.84	51.97	60.22	44.53	57.89
Contest	74.60	52.77	60.96	45.17	58.37
Library calls	73.72	51.01	61.22	43.68	57.41
Long-horizon	75.28	50.77	61.89	43.57	57.88

F Complete Ablation Studies

The results in Appendices F.1 to F.5 come from the controlled simulation of Appendix C.5: 256 instances per benchmark, four candidates, twelve inputs, a source with $\bar{\varepsilon} = 0.08$ and six requested calls, unless a row changes them. The subsections follow the ablation paragraphs of Section 5.3.

F.1 Update Rule and Source Error

The source-error rows of Table 4 are broken down by benchmark in Table 23 and Fig. 7. The soft update is at least as accurate as the vote on every benchmark and error rate, with three ties in 18 cells, and both stay above hard elimination. Hard elimination empties 9% to 15% of pools at $\bar{\varepsilon} = 0.02$ and 69% to 73% at $\bar{\varepsilon} = 0.2$, and the share of correct candidates it keeps stays near $(1-\varepsilon)^m$.

F.2 Query Rule and Update Rule

Each of the five query rules (Appendix C.7) is combined with the three update rules in Table 24. For the soft update and the vote, moving random calls into D helps, and the order by generalized binary search helps again, on all six benchmarks. Asking only between the two leading candidates matches this order on pools of four candidates. Hard elimination gains little from any query rule, as its accuracy is limited by wrong answers that remove correct candidates (Table 23).

F.3 Budget, Calibration Size and Query Order

With one requested call, the three update rules coincide (Table 25). From two calls on, the soft update and the vote rise until D runs out, while hard elimination peaks at two calls, where it keeps 86.5% of the correct candidates, and then declines. Table 26 varies the calibration size for three simulated sources; the vote and hard elimination do not use it and are constant. For

Table 20: DisJudge after four first-stage rules in the default setting (%). Accuracy of each rule alone and with DisJudge; increments δ over the count (69.87) of the rule and of both, where DisJudge alone adds 5.54; the additive prediction and its residual; and our re-run of the rule minus Table 2, in points and relative (Rel.).

First stage	Alone	With DisJudge	δ_{first}	δ_{both}	Additive	Residual	Re-run	Rel. (%)
CodeT	71.50	76.91	+1.63	+7.04	+7.17	−0.13	−0.14	−0.19
AlphaCode clustering	75.50	78.85	+5.63	+8.98	+11.17	−2.19	+0.13	+0.17
Execution self-consistency	73.20	77.46	+3.33	+7.59	+8.87	−1.28	−0.10	−0.14
S*	70.50	76.39	+0.63	+6.52	+6.17	+0.35	−0.07	−0.11

Table 21: Table 1 per benchmark (32B pools, %), with p_{other}; ρ is measured on HE+, LCB and CC only.

Benchmark	Level	μ	ρ	$p_\star$	p_{wrong}	p_{other}	a	h	Count	DisJudge	Gain
HE+	L1	+0.131	17.78	36.9	50.1	13.0	87.6	0.0101	69.87	75.41	+5.54
HE+	L2	+0.080	13.80	43.0	51.0	6.0	89.4	0.0034	70.86	74.72	+3.86
HE+	L3	+0.021	6.53	40.4	42.5	17.1	90.6	0.0003	72.47	73.66	+1.19
HE+	L4	−0.071	2.67	40.8	33.7	25.6	86.6	0.0034	73.90	74.35	+0.45
HE+	L5	−1.000	0.00	100.0	0.0	0.0	100.0	1	75.46	75.68	+0.22
MBPP+	L1	+0.128	–	36.7	49.5	13.8	87.6	0.0096	70.08	75.61	+5.53
MBPP+	L2	+0.078	–	38.5	46.3	15.2	87.3	0.0036	71.38	75.21	+3.83
MBPP+	L3	+0.021	–	41.2	43.3	15.5	87.5	0.0003	72.80	73.99	+1.19
MBPP+	L4	−0.077	–	43.1	35.4	21.5	88.2	0.0038	74.48	74.95	+0.47
MBPP+	L5	−1.000	–	100.0	0.0	0.0	100.0	1	75.79	76.02	+0.23
LCB	L1	+0.148	20.14	39.8	54.6	5.6	88.1	0.0117	46.43	52.37	+5.94
LCB	L2	+0.090	14.24	41.5	50.5	8.0	90.3	0.0044	47.58	51.71	+4.13
LCB	L3	+0.023	4.95	40.5	42.8	16.7	87.6	0.0003	49.24	50.55	+1.31
LCB	L4	−0.048	1.76	43.1	38.3	18.6	88.6	0.0014	50.61	51.13	+0.52
LCB	L5	−1.000	0.00	100.0	0.0	0.0	100.0	1	52.19	52.46	+0.27
BCB	L1	+0.149	–	35.5	50.4	14.1	89.9	0.0130	55.57	61.18	+5.61
BCB	L2	+0.091	–	37.1	46.2	16.7	89.5	0.0050	56.95	60.79	+3.84
BCB	L3	+0.024	–	41.3	43.7	15.0	89.1	0.0003	58.24	59.42	+1.18
BCB	L4	−0.059	–	41.6	35.7	22.6	86.9	0.0023	59.95	60.40	+0.45
BCB	L5	−1.000	–	100.0	0.0	0.0	100.0	1	61.33	61.55	+0.22
CC	L1	+0.124	17.73	37.6	50.0	12.5	83.7	0.0088	39.35	44.17	+4.82
CC	L2	+0.076	13.12	39.2	46.8	14.0	87.1	0.0034	40.46	43.83	+3.37
CC	L3	+0.020	5.71	44.8	46.8	8.4	86.8	0.0002	42.20	43.23	+1.03
CC	L4	−0.057	2.70	45.6	39.9	14.5	88.6	0.0019	43.46	43.88	+0.42
CC	L5	−1.000	0.00	100.0	0.0	0.0	100.0	1	45.08	45.28	+0.20
APPS	L1	+0.151	–	36.6	51.7	11.6	87.1	0.0130	45.07	51.48	+6.41
APPS	L2	+0.092	–	42.7	51.9	5.3	90.5	0.0045	46.29	50.72	+4.43
APPS	L3	+0.024	–	42.5	44.9	12.6	90.4	0.0003	47.62	49.03	+1.41
APPS	L4	−0.066	–	45.6	39.0	15.4	88.1	0.0026	49.31	49.86	+0.55
APPS	L5	−1.000	–	100.0	0.0	0.0	100.0	1	50.72	51.01	+0.29

the default source S1 the soft update is within 0.5 points of its value under the true error rates from 16 records on. For the weaker source S3, 16 records leave it below the vote, and from 64 records on it is above. With 64 or more records, S1 and S2 do as well as with the true rates.

A query order computed once at the start (fixed) is compared with one recomputed after every answer (adaptive) in Table 27 and Fig. 8. Recomputing helps the vote and the soft update on every benchmark, by 3.1 to 6.6 points, and hard elimination on four of the six, since a removed candidate cannot be reconsidered (Table 28). Table 29 compares g computed by rule from the outputs and the weights with g predicted by a model. The rule is more accurate for all three update rules and takes 0.09 ms per instance against 689 ms for the same 39.2 evaluations, since it computes g exactly.

F.4 First-Stage Rules and Simulated Sources

In front of the simulation (Table 30), every first-stage rule raises the soft update and the vote above their values without one, and with one in front the vote leads the soft update by 2.1 to 2.6 points on these pools of four. For the five simulated sources of Table 31, the last column gives the error per call implied by the share of correct candidates that hard elimination keeps, $1 - \text{kept}^{1/m}$: 8.5% for S1, 8.8% for S2, 17.2% for S3 and 31.9% for S4. The soft update is the most accurate rule for S1 to S4, and its lead over the vote grows with this error, from 0.9 points for S1 to 40.3 points for S4. With the exact reference S5 all three update rules coincide.

Against the default independent source (Table 32), the correlated source S4 is less accurate overall and much less so on D, and worse on D for 80% to 87% of the instances. Under it the vote remains

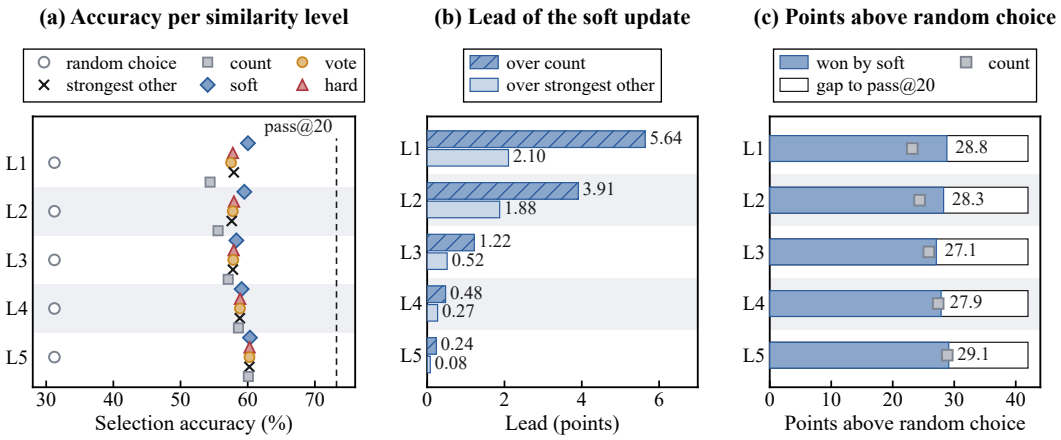

Figure 6: Rules along the similarity levels (32B pools, six-benchmark means; values in Table 22). (a) Accuracy of random choice, the count, the three update rules of DisJudge and the strongest other rule of Table 2, with the pass@20 upper bound. (b) Lead of the soft update over the count and the strongest other rule. (c) Points won by the soft update and by the count out of the 42.0 points between random choice and the upper bound.

Table 22: Rules along the similarity levels (32B pools, six-benchmark means, %): accuracy of random choice, the count, the three update rules and the strongest other rule; the lead of the soft update over the count and over that rule; and the share of the gap between the count and pass@20 (73.22%) that the soft update closes.

	Selection accuracy						Soft update		
Level	Random	Count	Vote	Hard	Soft	Best other	− count	− best other	Gap closed (%)
L1	31.23	54.40	57.52	57.79	60.04	57.93	+5.64	+2.10	30.0
L2	31.23	55.59	57.78	57.95	59.50	57.62	+3.91	+1.88	22.2
L3	31.23	57.09	57.85	57.91	58.31	57.80	+1.22	+0.52	7.6
L4	31.23	58.62	58.84	58.86	59.10	58.83	+0.48	+0.27	3.3
L5	31.23	60.09	60.26	60.28	60.33	60.25	+0.24	+0.08	1.8

near a quarter for any number of calls and hard elimination loses almost all correct candidates, while the soft update gains with each further call until D runs out, reaching 69.5% at twelve calls.

F.5 Reference Coverage and Pools Without a Correct Candidate

We vary the share of queries that the exact reference covers and split the instances by the error type of their wrong candidates (Table 33, Fig. 9). The three update rules select alike, since the reference never errs where it answers. Accuracy grows with coverage for arithmetic, off-by-one and logic errors, which reach 98.3% to 99.1% at full coverage. Overflow and complexity errors lie beyond the size bound of the reference, so the accuracy remains 23.3% and 27.1% at all coverages.

The full sweep behind Fig. 9 is in Table 34. For the three error types that the reference can reach (arithmetic, off-by-one and logic), the covered share and the accuracy both rise with coverage.

On pools that hold no correct candidate (Table 35, Fig. 10), no rule can succeed, and all three update rules score zero. Hard elimination empties 90% to 95% of these pools, while the other two rules keep every candidate (Table 35). The calls are spent before it is known whether a correct candidate exists: such an instance uses 5.95 calls, almost as many as the other instances.

F.6 Per-Benchmark Gains and Calls to a Target

Fig. 11 returns to the full pools. Panel (a) and Table 37 show the accuracy of the count and of DisJudge on the 32B pools. The accuracy rises by 4.8 to 6.4 points while μ of each benchmark stays the same, since DisJudge keeps the candidates, the inputs and the oracle and only reweighs their answers. Panel (b) and Table 38 give the reliable calls per instance that the vote and hard elimination need to reach a fixed target accuracy as T grows. Hard elimination reaches the target with 4 calls, from $T = 50$ on HumanEval+ and LiveCodeBench and from $T = 25$ on CodeContests, and the vote needs 8 or 9 calls and more tests; in 13 of the 30 cells the target is

Table 23: Three update rules at three source error rates and six calls (simulation, %): accuracy per benchmark and its mean; the mean share of correct candidates kept and of pools emptied by each update rule.

Update	$\bar{\varepsilon}$	HE+	MBPP+	LCB	BCB	CC	APPS	Mean	Kept	Emptied
hard	0.02	89.1	88.3	84.8	83.2	85.2	87.9	86.39	87.30	11.98
hard	0.08	59.4	58.6	56.3	52.0	59.4	62.9	58.07	58.59	38.74
hard	0.2	22.3	25.0	23.4	23.0	27.0	27.0	24.61	24.93	71.22
soft	0.02	98.4	98.0	99.2	95.7	98.4	98.8	98.11	100.00	0.00
soft	0.08	93.0	92.2	93.8	92.2	94.1	94.1	93.23	100.00	0.00
soft	0.2	69.9	76.2	78.9	76.2	82.0	76.2	76.56	100.00	0.00
vote	0.02	97.7	97.7	98.8	94.9	98.4	98.8	97.72	100.00	0.00
vote	0.08	93.0	91.4	92.6	90.2	93.4	93.4	92.32	100.00	0.00
vote	0.2	66.8	72.3	73.8	71.9	75.0	74.2	72.33	100.00	0.00

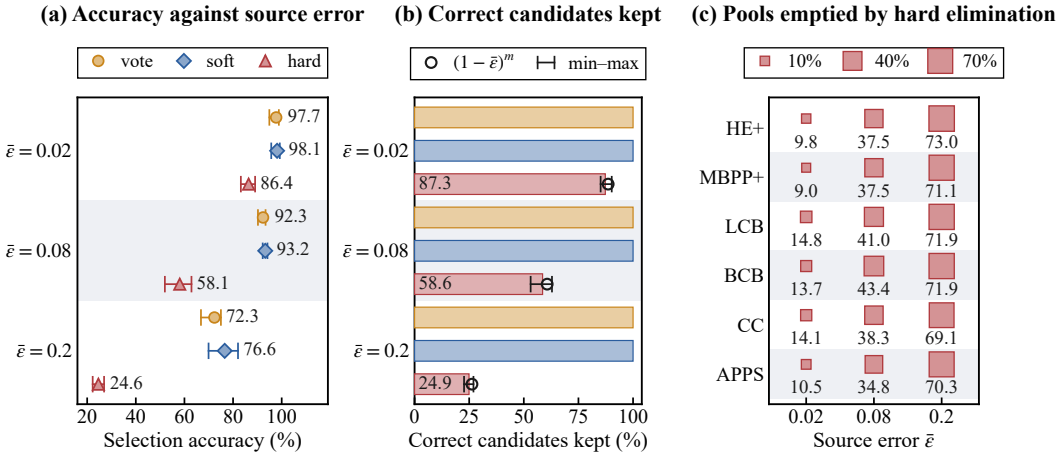

Figure 7: Three update rules against the source error. (a) Selection accuracy, six-benchmark mean with its range. (b) Share of correct candidates kept, with its range and $(1-\bar{\varepsilon})^m$ at $m = 5.99$ calls. (c) Share of pools that hard elimination empties per benchmark; the vote and the soft update never empty a pool.

not reached, because D is too small at low T. Panel (c) plots the accuracy of the three update rules against the requested calls in the simulation, for the independent and the correlated source.

G Denominators of the Rates

The denominators of the safety rates at the default pool of each benchmark are in Table 39: the 20 candidates of each of the 4,096 instances, split into correct and wrong ones, the instances with an empty D, which make no call, and the removals of the soft update, of which there are none, so that its elimination precision is undefined. The shares $p_\star$, p_{wrong} and p_{other} are pooled over the pairs of a benchmark, and each selection accuracy is a share of the instances of its pool.

Table 24: Five query rules crossed with three update rules (controlled simulation, %): accuracy per benchmark and its mean; mean share of correct candidates kept and of pools emptied by each update rule.

Update	Query rule	HE+	MBPP+	LCB	BCB	CC	APPS	Mean	Kept	Emptied
hard	Random over all inputs	55.5	55.5	48.4	51.2	53.1	57.0	53.45	59.24	36.65
hard	Random within D	57.8	57.8	55.5	50.8	57.0	60.9	56.64	58.85	38.80
hard	GBS order within D	59.4	58.6	56.3	52.0	59.4	62.9	58.07	58.59	38.74
hard	Margin order within D	59.0	58.6	55.9	52.3	59.0	62.9	57.94	58.46	39.19
hard	Top-2 order within D	59.8	58.2	56.6	52.7	58.6	63.3	58.20	58.73	38.51
soft	Random over all inputs	77.7	76.6	78.9	78.1	78.9	81.3	78.58	100.00	0.00
soft	Random within D	84.0	84.8	87.1	85.9	89.5	89.5	86.78	100.00	0.00
soft	GBS order within D	93.0	92.2	93.8	92.2	94.1	94.1	93.23	100.00	0.00
soft	Margin order within D	88.7	87.9	92.2	89.8	94.5	92.6	90.95	100.00	0.00
soft	Top-2 order within D	92.2	92.2	93.8	92.2	95.3	94.1	93.29	100.00	0.00
vote	Random over all inputs	78.1	78.9	78.9	75.0	79.7	78.5	78.19	100.00	0.00
vote	Random within D	84.8	86.3	88.3	82.0	89.8	86.3	86.26	100.00	0.00
vote	GBS order within D	93.0	91.4	92.6	90.2	93.4	93.4	92.32	100.00	0.00
vote	Margin order within D	91.4	88.7	91.8	88.3	93.0	90.6	90.62	100.00	0.00
vote	Top-2 order within D	92.6	91.8	93.0	91.0	93.8	93.4	92.58	100.00	0.00

Table 25: Requested calls per instance (six-benchmark means, %): calls made, capped by N_D; accuracy of the three update rules; share of correct candidates kept and of pools emptied by hard elimination.

Requested	Made	Vote	Hard	Soft	Kept	Emptied
1	1.00	44.27	44.27	44.27	91.34	0.00
2	2.00	81.12	61.25	81.18	86.50	2.40
3	3.00	82.23	60.08	83.66	82.70	5.90
6	5.99	92.32	58.07	93.23	58.59	38.74
9	8.43	93.62	52.15	94.14	52.48	44.82
12	9.03	93.62	47.66	94.21	47.92	50.71

Table 26: Calibration records per query type for sources S1 to S3 (six-benchmark means, %): accuracy of the three update rules with error rates estimated from 16, 64 or 128 records, or set to their true values.

Source	Records	Vote	Hard	Soft
S1	16	92.32	58.07	92.77
S1	64	92.32	58.07	93.23
S1	128	92.32	58.07	93.23
S1	true	92.32	58.07	93.16
S2	16	88.93	56.71	90.04
S2	64	88.93	56.71	91.73
S2	128	88.93	56.71	91.86
S2	true	88.93	56.71	91.60
S3	16	69.99	31.84	67.90
S3	64	69.99	31.84	72.46
S3	128	69.99	31.84	73.76
S3	true	69.99	31.84	74.09

Table 27: Fixed and adaptive query order (simulation, six calls, %): accuracy of the three update rules per benchmark and its mean, with the query order computed once (off) or recomputed after every answer (on).

Update	Order	HE+	MBPP+	LCB	BCB	CC	APPS	Mean
hard	off	60.2	55.9	57.0	51.2	56.6	62.5	57.23
hard	on	59.4	58.6	56.3	52.0	59.4	62.9	58.07
soft	off	88.7	85.9	89.1	87.1	90.2	90.2	88.54
soft	on	93.0	92.2	93.8	92.2	94.1	94.1	93.23
vote	off	87.5	85.9	89.5	83.6	89.1	89.1	87.43
vote	on	93.0	91.4	92.6	90.2	93.4	93.4	92.32

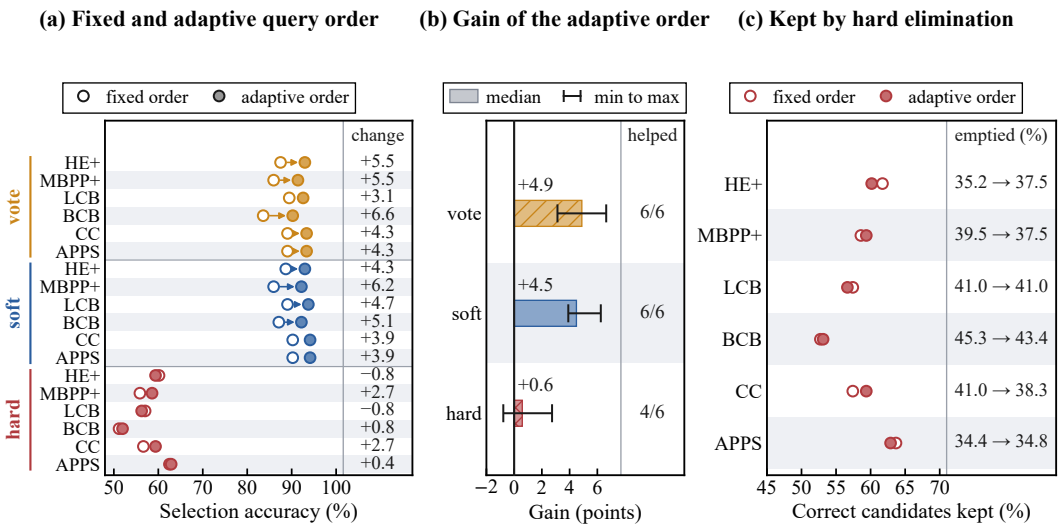

Figure 8: Fixed and adaptive query order. (a) Accuracy per benchmark with the fixed (hollow) and adaptive (filled) order for the three update rules. (b) Median gain over benchmarks, its range, and the benchmarks helped. (c) Correct candidates kept by hard elimination with the two orders, and the change in emptied pools.

Table 28: Hard elimination with the fixed (off) and the adaptive (on) order (simulation, six calls, %): share of correct candidates kept and of pools emptied per benchmark; these are the values behind Fig. 8(c).

	Kept		Emptied	
Benchmark	off	on	off	on
HE+	61.72	60.16	35.16	37.50
MBPP+	58.59	59.38	39.45	37.50
LCB	57.42	56.64	41.02	41.02
BCB	52.73	53.12	45.31	43.36
CC	57.42	59.38	41.02	38.28
APPS	63.67	62.89	34.38	34.77

Table 29: Computing g by rule or by model (simulation, %): accuracy, evaluations of g and time per instance.

Update	g by	HE+	MBPP+	LCB	BCB	CC	APPS	Mean	Evaluations	Time (ms)
hard	rule	59.4	58.6	56.3	52.0	59.4	62.9	58.07	39.17	0.090
hard	model	47.3	44.9	42.6	40.6	40.2	52.0	44.60	39.17	689.323
soft	rule	93.0	92.2	93.8	92.2	94.1	94.1	93.23	39.17	0.090
soft	model	89.5	90.6	92.2	87.5	93.4	91.4	90.76	39.17	689.323
vote	rule	93.0	91.4	92.6	90.2	93.4	93.4	92.32	39.17	0.090
vote	model	89.8	89.8	91.0	87.9	91.8	92.2	90.43	39.17	689.323

Table 30: First-stage rules in front of the simulation (accuracy, six-benchmark means, %).

First stage	Vote	Hard	Soft
None	92.32	58.07	93.23
AlphaCode clustering	95.84	58.22	93.42
Execution self-consistency	96.16	58.51	93.55
Reranker	95.93	58.84	93.62
S*	96.34	59.26	94.21

Table 31: Five simulated sources (six-benchmark means, %): calls made, accuracy of the three update rules, share of correct candidates kept and of pools emptied by hard elimination, and the error per call implied by the kept share. S1 is the default, S2 and S3 err more on D, S4 is correlated; S5 answers only the queries it covers.

Source	Calls	Vote	Hard	Soft	Kept	Emptied	Error per call
S1	5.99	92.32	58.07	93.23	58.59	38.74	8.5
S2	5.99	88.93	56.71	91.86	57.42	38.67	8.8
S3	5.99	69.99	31.84	73.76	32.36	58.20	17.2
S4	5.99	28.45	9.70	68.75	10.03	60.35	31.9
S5	4.80	84.05	84.05	84.05	100.00	0.00	0.0

Table 32: Independent against correlated source (%). Top: overall accuracy and accuracy on D of the two sources, the share of instances where the correlated source is worse on D, and the accuracy of the three update rules at six calls, per benchmark. Bottom: the correlated source by requested calls from 1 to 12, six-benchmark means, with the share of correct candidates kept and of pools emptied by hard elimination.

	Independent		Correlated			Independent, 6 calls			Correlated, 6 calls		
	Overall	on D	Overall	on D	Worse	Vote	Hard	Soft	Vote	Hard	Soft
HE+	85.8	91.9	80.2	68.3	85.9	93.0	59.4	93.0	25.8	9.0	68.4
MBPP+	85.7	91.8	81.0	69.3	84.8	91.4	58.6	92.2	32.8	9.4	64.8
LCB	85.7	91.3	78.8	68.0	85.2	92.6	56.3	93.8	26.6	9.4	72.3
BCB	85.5	91.0	80.1	68.7	79.7	90.2	52.0	92.2	28.5	11.7	65.6
CC	85.6	91.1	79.4	69.5	82.0	93.4	59.4	94.1	28.9	9.8	70.7
APPS	86.1	92.3	80.5	67.9	86.7	93.4	62.9	94.1	28.1	9.0	70.7

Requested	Vote	Hard	Soft	Kept	Emptied
1	26.63	26.63	38.02	56.58	0.00
2	28.71	32.16	54.30	33.07	0.07
3	29.49	24.54	59.57	25.00	19.40
6	28.45	9.70	68.75	10.03	60.35
9	26.76	4.04	69.40	4.36	76.89
12	26.50	3.71	69.47	4.04	79.04

Table 33: Reference coverage by error type and setting (%): accuracy, reliable calls made, the share of errors of that type the reference covers, and the number of instances and of covered instances over the six benchmarks.

Error type	Setting	Accuracy	Calls	Covered share	Instances	Covered
all	0.2	32.89	0.70	11.65	1536	179
all	0.4	41.96	1.43	23.83	1536	366
all	0.6	51.36	2.18	36.26	1536	557
all	0.8	60.10	2.87	47.79	1536	734
all	1	69.22	3.59	59.90	1536	920
arithmetic	0.2	38.41	1.24	20.63	315	65
arithmetic	1	99.05	5.98	100.00	315	315
off by one	0.2	38.26	1.15	19.13	298	57
off by one	1	98.32	5.98	100.00	298	298
logic	0.2	37.46	1.12	18.57	307	57
logic	1	98.70	6.00	100.00	307	307
overflow	0.2 to 1	23.29	0.00	0.00	308	0
complexity	0.2 to 1	27.11	0.00	0.00	308	0

Table 34: Full coverage sweep (%): accuracy and the share of each error type that the reference covers.

	Selection accuracy at coverage					Share of the type covered				
Error type	0.2	0.4	0.6	0.8	1	0.2	0.4	0.6	0.8	1
all	32.89	41.96	51.36	60.10	69.22	11.7	23.8	36.3	47.8	59.9
logic	37.46	52.27	66.27	83.67	98.70	18.6	39.0	58.4	79.0	100.0
off by one	38.26	56.81	71.35	84.76	98.32	19.1	43.7	64.7	83.5	100.0
arithmetic	38.41	50.65	69.08	82.04	99.05	20.6	36.8	58.6	77.0	100.0
overflow	23.29	23.29	23.29	23.29	23.29	0.0	0.0	0.0	0.0	0.0
complexity	27.11	27.11	27.11	27.11	27.11	0.0	0.0	0.0	0.0	0.0

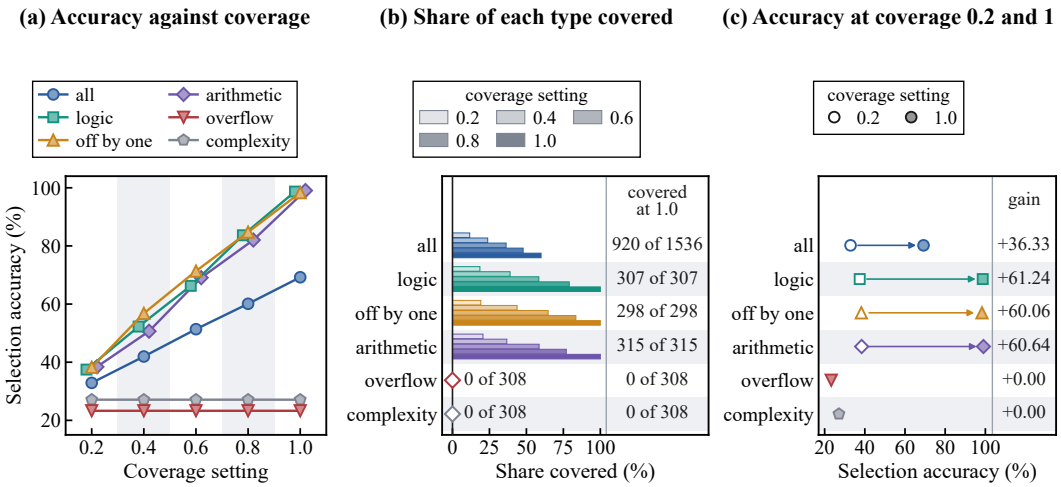

Figure 9: Reference coverage by error type. (a) Accuracy against the coverage setting, one line per error type and one for all. (b) Share of each type covered per setting. (c) Accuracy at coverage 0.2 and 1, with the gain.

Table 35: Pools without a correct candidate (six-benchmark means, %): accuracy per update rule, share of pools emptied with its range over benchmarks, calls made, and, for comparison, accuracy on pools that hold one.

		Pools emptied				With a correct
Update	Accuracy	Mean	Min	Max	Calls	candidate
hard	0.00	92.38	90.23	95.31	5.949	58.07
soft	0.00	0.00	0.00	0.00	5.949	93.23
vote	0.00	0.00	0.00	0.00	5.949	92.32

Table 36: Pools without a correct candidate, per benchmark (%): accuracy of the three update rules when the pool holds one, share of such pools emptied by hard elimination, and calls made on both kinds of pools.

	Accuracy, pool holds one				Calls made	
Benchmark	Vote	Hard	Soft	Emptied (none)	none	one
HE+	92.97	59.38	92.97	91.41	5.949	5.984
MBPP+	91.41	58.59	92.19	91.41	5.957	5.988
LCB	92.58	56.25	93.75	95.31	5.969	5.984
BCB	90.23	51.95	92.19	90.23	5.938	5.988
CC	93.36	59.38	94.14	92.97	5.938	5.996
APPS	93.36	62.89	94.14	92.97	5.945	5.984

Table 37: Accuracy before and after DisJudge (32B pools, %): the count, DisJudge with the soft update, the gain, and the bias μ of the oracle, which is the same before and after because the oracle does not change.

Benchmark	Count	DisJudge	Gain	μ
HE+	69.87	75.41	+5.54	+0.131
MBPP+	70.08	75.61	+5.53	+0.128
LCB	46.43	52.37	+5.94	+0.148
BCB	55.57	61.18	+5.61	+0.149
CC	39.35	44.17	+4.82	+0.124
APPS	45.07	51.48	+6.41	+0.151

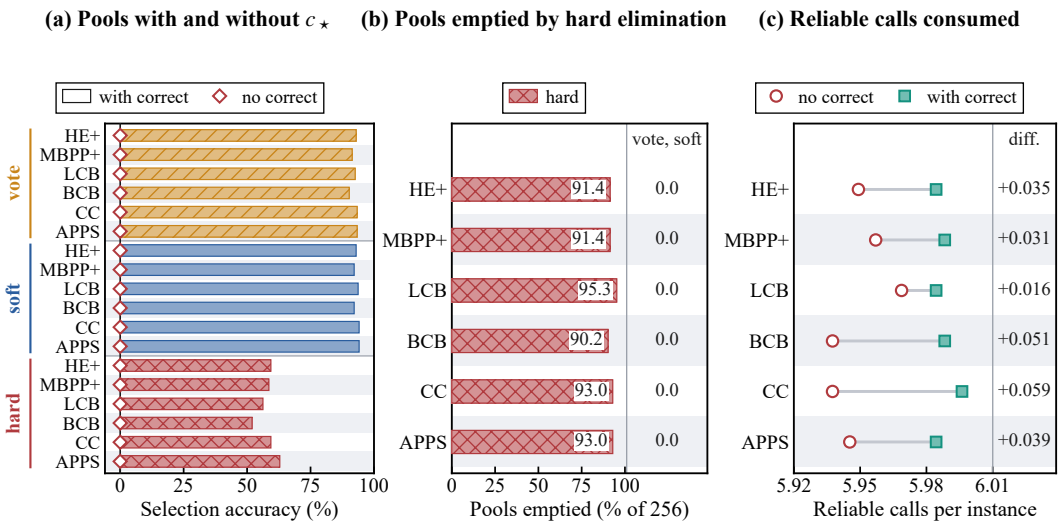

Figure 10: Pools without a correct candidate. (a) Accuracy per benchmark when the pool holds a correct candidate (bars) and when it holds none (diamonds at zero). (b) Share of such pools emptied by hard elimination. (c) Reliable calls made per instance on pools without and with a correct candidate, per benchmark.

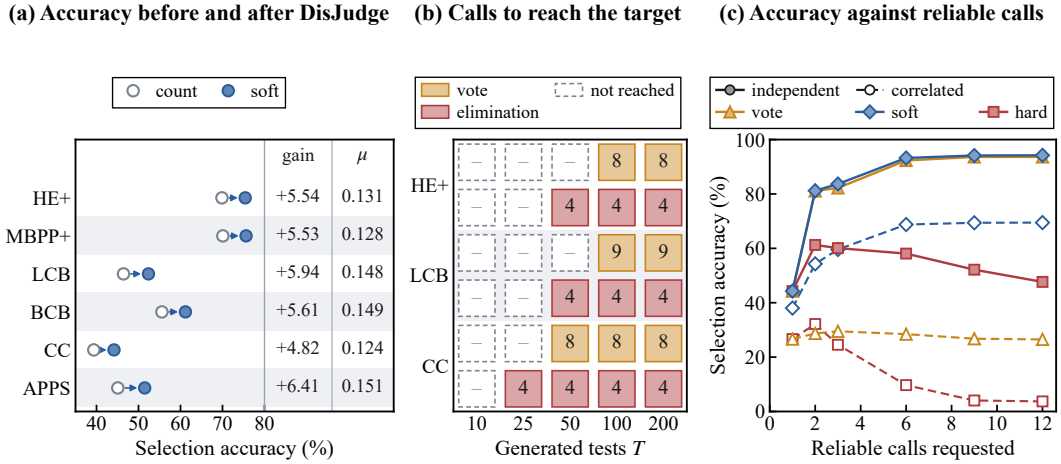

Figure 11: Before and after, calls to a target, and calls in the simulation. (a) Selection accuracy of the count and of DisJudge per benchmark (32B pools), with the bias μ of each benchmark, which does not change. (b) Reliable calls per instance needed to reach a fixed target accuracy, for the vote and hard elimination at five values of T from 10 to 200; dashed cells are not reached. (c) Accuracy of the three update rules against the requested calls, from 1 to 12, for the independent source (solid) and the correlated source (dashed).

Table 38: Calls to a target and the bias sweep. Left: reliable calls per instance to reach a fixed target accuracy on HE+, LCB and CC, for the vote and hard elimination at each T; a dash means that it is not reached. Right: the bias sweep of Fig. 2(a), with μ and the gain of the soft update over the count at the low, mid and high bias levels.

		Tests T				
Benchmark	Update	10	25	50	100	200
HE+	vote	–	–	–	8	8
HE+	hard	–	–	4	4	4
LCB	vote	–	–	–	9	9
LCB	hard	–	–	4	4	4
CC	vote	–	–	8	8	8
CC	hard	–	4	4	4	4

	μ			Gain (points)		
Benchmark	Low	Mid	High	Low	Mid	High
HE+	−0.064	+0.026	+0.151	0.20	1.21	5.86
MBPP+	−0.062	+0.011	+0.142	0.48	0.66	5.50
LCB	−0.078	+0.014	+0.143	0.22	0.86	5.85
BCB	−0.058	+0.018	+0.138	0.21	1.09	4.99
CC	−0.074	+0.019	+0.143	0.34	1.10	5.04
APPS	−0.061	+0.026	+0.150	0.28	1.20	5.45

Table 39: Denominators at the default pool of each benchmark: correct and wrong candidates over the $4{,}096$ instances, candidates removed by the soft update, $\Pr(N_D = 0)$ and the share of instances with no call (%), coverage of D by the reference (%), and the number of instances whose elimination precision has a zero denominator.

Benchmark	Correct	Wrong	Instances	Removed by soft	$\Pr(N_D{=}0)$	No call	Ref. cov.	Zero denominators
HE+	30547	51373	4096	0	1.93	1.93	86.68	4096
MBPP+	36364	45556	4096	0	1.87	1.87	87.62	4096
LCB	20820	61100	4096	0	1.28	1.28	85.70	4096
BCB	26714	55206	4096	0	1.27	1.27	80.52	4096
CC	20076	61844	4096	0	1.42	1.42	78.52	4096
APPS	18955	62965	4096	0	1.94	1.94	80.61	4096