Carnegie Mellon University

Language Technologies Institute

PACE: A Proxy for Agentic Capability Evaluation

Yueqi Song[1], **Lintang Sutawika**[1], **Jiarui Liu**[1], **Lindia Tjuatja**[1], **Jiayi Geng**[1], **Yunze Xiao**[1], **Daniel Lee**[2], **Aditya Bharat Soni**[1], **Vincent Lo**[1], **Xiang Yue**[1], **Graham Neubig**[1]

[1]Carnegie Mellon University [2]Salesforce AI Research

{yueqis, gneubig}@cs.cmu.edu

neulab/pace neulab/pace-bench

Abstract

Evaluating large language model (LLM) agents on benchmarks like SWE-Bench and GAIA can be expensive, time-consuming, and requires complex infrastructure. A single evaluation can cost thousands of dollars and take days to complete. In contrast, non-agentic LLM benchmarks that test individual capabilities (e.g., reasoning, code generation, instruction following) are fast and cheap to run. In this paper, we investigate whether performance on expensive agentic benchmarks can be accurately predicted by the performance on a small, carefully selected subset of atomic evaluation instances. We introduce PACE, a framework that constructs proxy benchmarks by selecting instances from existing non-agentic evaluations whose aggregate scores most reliably predict model performances on agentic benchmarks. Given a pool of candidate instances spanning atomic capabilities (instruction following, planning, tool calling, etc.), PACE fits a regression that maps a model's scores on a compact subset of source instances to its score on the target agentic benchmark. The subset itself is curated by combining two complementary instance-selection strategies, target-relevance local selection and globally informative global selection. We apply PACE to the 4 target agentic benchmarks in this paper, which yields PACE-BENCH, the concrete proxy benchmark that we evaluate in the paper. Experiments across 14 models, 4 agentic benchmarks, and 19 non-agentic benchmarks show that PACE-BENCH predicts agentic scores with leave-one-out cross-validation (LOOCV) mean absolute error (MAE) under 4%, Spearman correlation above 0.80, and pairwise model-ranking accuracy around 85%, all at much less than 1% of the full agentic evaluation cost. We further analyze the selected proxy instances, revealing which skills each agentic benchmark uniquely demands. PACE enables practitioners to obtain reliable estimates of agentic performance during model development, selection, and routing, without the overhead of full agent evaluation.

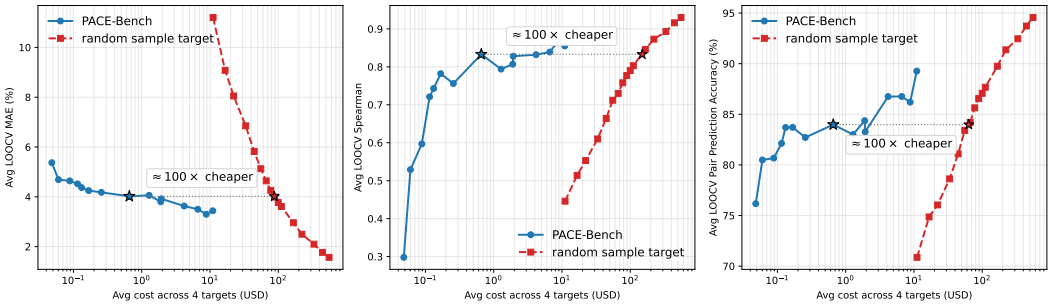

Figure 1: **Cost-versus-quality tradeoff** of PACE (blue) and sub-sampling target agentic evals (red), averaged across four datasets. **Left:** mean absolute error. **Middle:** Spearman correlation. **Right:** pairwise model-ranking accuracy. At every budget below saturation, PACE dominates sub-sampling agentic evals on all three metrics, matching quality at roughly $1/100$ of the cost.

Submission in Progress.

1 Introduction

Tracking the progress of large language models (LLMs) capabilities has long relied on fast and inexpensive benchmarks that evaluate models' individual capabilities such as knowledge retrieval (Hendrycks et al., 2021a; Wang et al., 2024b), mathematical reasoning (Hendrycks et al., 2021b), instruction following (Zhou et al., 2023), code generation (Jain et al., 2024), and more. Because such benchmarks consist of short, self-contained instances that can be scored with a single model invocation, these instances are cheap to run, easy to reproduce, and widely used for informing model development (Liang et al., 2023; Biderman et al., 2024).

As language models are increasingly deployed as agents, however, this evaluation paradigm breaks down. Agentic benchmarks such as SWE-Bench (Jimenez et al., 2024), GAIA (Mialon et al., 2024), and WebArena (Zhou et al., 2024) require models to operate over long horizons, interact with tools or environments, and recover from errors (Wang et al., 2024a), often requiring complex infrastructure, long rollout times, and substantial API costs. Even evaluation of a single model under a single agent harness could cost thousands of dollars and take hours or days of setup and execution. These burdens often force researchers to evaluate models less frequently, report results on limited subsets, and make rigorous agent evaluation disproportionately accessible only to well-resourced groups.

Despite the complexity of agentic tasks, success of LLMs on agentic benchmarks depends on model abilities like instruction following, planning, tool use, and reasoning, which are already measured by fast and inexpensive non-agentic benchmarks (Sumers et al., 2023; Xi et al., 2025). However, researchers still run full agent evaluations to compare models to track progress, suggesting that the predictive connection between model performance on non-agentic and agentic tasks is not yet well understood. Thus, we ask: *can non-agentic benchmarks serve as a reliable and low-cost proxy for agentic benchmarks*?

To answer this question, we propose PACE (Proxy for Agentic Capability Evaluation), a simple yet effective framework that selects a compact subset of non-agentic benchmark instances whose aggregate scores could best predict the target agentic benchmark performances across models. PACE draws its candidate pool of non-agentic evaluation instances from existing benchmarks that broadly cover skills that intuitively seem important for agentic tasks, such as instruction following, tool calling, multimodal understanding, etc. We formulate the construction of PACE as a budget-constrained subset selection problem. Given a fixed budget of C proxy instances, PACE uses a calibration set of models with known scores on both the candidate pool and a target agentic benchmark to identify which C instances are the most predictive of the target benchmark. Concretely, PACE fits a least-squares regression that maps each model's per-instance scores on the C selected source instances to its target benchmark mean, with the calibration models supplying training data; bootstrap resampling over the target instances stabilizes the regression weights against label noise. The C instances themselves are produced by combining two complementary criteria, a target-relevance local signal (rank-correlation with target labels) and a globally-informative global signal (SVD leverage in the source matrix).

Our approach differs from prior benchmark compression and subset-selection methods (Perlitz et al., 2024; Polo et al., 2024), which aim to reduce costs within a single target benchmark, and from approaches that recast agent tasks into alternative formats (e.g., multiple choice questions) (Qin et al., 2025). Instead, we seek to predict model performances on a target agentic benchmark using a compact subset drawn from a separate candidate pool of inexpensive evaluation instances, with no modification to how the target benchmark is scored.

To demonstrate the empirical effectiveness of PACE, we evaluate PACE across 14 models, 4 agentic benchmarks (GAIA (Mialon et al., 2024), SWE-Bench Multimodal (Yang et al., 2025), SWE-Bench Verified (Jimenez et al., 2024), SWT-Bench (Mündler et al., 2024)), and 19 source non-agentic benchmarks spanning 11 capabilities of LLMs.

Figure 1 summarizes our answer: a small, well-chosen subset of non-agentic instances tracks agentic performance closely, at less than $\frac{1}{100}$ of the cost of either a full agent evaluation or a random subset of the target benchmark itself. Concretely, using a proxy of just 100 instances, PACE achieves strong predictive performance. At equal prediction quality, PACE requires roughly $100\times$ less cost in dollars than a random target-sampling baseline. Our main findings are as follows:

- **Generalization to Unseen Models:** Instances selected with PACE on a training set of models generalize to a held-out set of models not seen during selection. This setting is relevant to

Benchmark	IF	LCA	ER	Plan	Code	IR	CS	TC	Reas	MM	Ver	Setup	#Inst	Cost
Agentic Benchmarks														
GAIA	•	•	•	•	•	•		•	•	•	•	H	165	$ 0.38
SWE-Bench Multimodal	•	•	•	•	•		•	•	•	•	•	H	102	$ 1.89
SWE-Bench Verified	•	•	•	•	•		•	•	•		•	H	500	$ 1.19
SWT-Bench	•	•	•	•	•		•		•		•	H	430	$ 0.98
Non-Agentic Benchmarks														
ACPBench	•			•					•		•	L	1,040	$ 0.009
AIME 2025	•								•			L	30	$ 0.015
BEIR (NFCorpus)	•					•						L	323	$ 0.121
BFCL	•							•	•		•	L	5,343	$ 0.008
DebugBench	•		•		•				•		•	M	4,253	$ 0.005
GPQA	•								•			L	2,384	$ 0.009
HumanEval	•				•				•		•	L	164	$ 0.004
IFEval	•								•			L	541	$ 0.006
InFoBench	•								•			L	500	$ 0.009
LIFBench	•	•							•			L	2,766	$ 0.118
LiveCodeBench	•		•	•	•				•		•	M	2,870	$ 0.051
LogiQA	•								•			L	1,302	$ 0.007
MBPP	•				•				•		•	L	500	$ 0.004
MMLU	•								•			L	28,084	$ 0.006
MMMU	•								•	•		L	900	$ 0.012
PlanBench	•			•					•		•	L	4,000	$ 0.007
RepoBench-R	•	•			•		•		•			M	2,010	$ 0.013
VisualPuzzles	•								•	•		L	1,168	$ 0.018
VisualWebBench	•					•			•	•		L	1,536	$ 0.007

Table 1: Overview of existing agentic and non-agentic benchmarks, with capability coverage, setup complexity, number of instances, and estimated per-instance evaluation cost for Claude Sonnet 4.5. **IF**=Instruction Following, **LCA**=Long Context Aggregation, **ER**=Error Recovery, **Plan**=Planning, **Code**=Code Generation, **IR**=Information Retrieval, **CS**=Code Search, **TC**=Tool Calling, **Reas**=Reasoning, **MM**=Multimodal Understanding, **Ver**=Verification and Test. We classify a setup effort for each benchmark within 3 classifications of **High** (requiring setting up evaluation environments), **Medium** (medium effort setup) or **Low** (only needing API calls).

deployment, where the goal is to predict a new model's agentic performance, without committing to full agentic evaluation. Specifically, across the 4 benchmarks, PACE predicts agentic scores with leave-one-out cross-validation (LOOCV) mean absolute error (MAE) under 4%, Spearman correlation above 0.80, and pairwise model-ranking accuracy around 85%, all much less than 1% of the full agentic evaluation cost.

- **Predictable Cost-Accuracy Tradeoff:** We show that the cost-accuracy tradeoff is smooth and highly predictable. As shown in Figure 1, prediction quality broadly improves with the proxy budget and then saturates, with diminishing returns past a few hundred instances; even small budgets are already highly competitive. This lets practitioners choose evaluation budgets that match their specific resource constraints.
- **Interpretability of Capabilities:** By examining the number of selected instances from each benchmark and each model capability, we identify which model capabilities most strongly affect each target agentic task, providing interpretable evidence for the capability structure that underlies successes and failures in agentic tasks.

2 Background

2.1 From Static to Agentic Benchmarks

Standard LLM evaluation has historically relied on atomic, single-turn, and largely static benchmarks that are inexpensive to run and easy to reproduce. In contrast, evaluating LLM-based agents requires models to act over longer horizons, interact with tools or environments, and recover from intermediate errors. The agentic benchmarks often require sandboxes, browsers, repositories, or custom runtime environments, and can be sensitive to environmental noise, harness design, and external dependencies (Fan et al., 2025). What makes prediction plausible across these two evaluation protocols is

that they require and evaluate models on overlapping underlying capabilities. We organize these into 11 categories: instruction following, long context aggregation, error recovery, planning, code generation, information retrieval, code search, tool calling, reasoning, multimodal understanding, and verification and test[1]. Table 1 maps each non-agentic and agentic benchmark to the capabilities it requires, alongside its evaluation cost and setup requirements. SWE-Bench (Jimenez et al., 2024), for instance, requires models to possess capabilities like planning, code generation, and verification, all of which are also measured by cheaper non-agentic benchmarks, suggesting its agentic performance is in principle predictable from non-agentic signals.

Recent work suggests that complex model-level behavior can be partially predicted from simpler evaluation signals. Collaborative Performance Prediction (CPP) (Zhang et al., 2024) uses matrix factorization to estimate missing model-task outcomes, ONEBench (Ghosh et al., 2025) studies sample-level unification across open-ended capabilities, and several meta-analyses have shown that aggregated static benchmark scores can predict human-preference-based Elo (Spangher et al., 2025; Ramaswamy et al., 2025). Although human preference and agentic execution are substantively different outcomes, this literature supports the broader premise that model performance exhibits predictable structure across evaluations. This motivates the question: if agentic success depends on model capabilities already measured by cheaper non-agentic benchmarks, a carefully selected subset of those instances may serve as a reliable proxy.

2.2 Problem Formulation

Let $M = \{m_1, \ldots, m_{|M|}\}$ be a set of $|M|$ language models and $T = \{t_1, \ldots, t_{|T|}\}$ be a target agentic benchmark with $|T|$ instances. For each model m and index $i \in \{1, \ldots, |T|\}$, let $y_{m,i} \in [0, 1]$ denote m's score on instance $t_i \in T$. Collecting over all models and instances gives the target score matrix $Y \in [0, 1]^{|M| \times |T|}$. We write $\bar{y}_m = \frac{1}{|T|} \sum_{i=1}^{|T|} y_{m,i}$ for model m's mean score on the target benchmark. Let also $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$ be a set of $|\mathcal{S}|$ non-agentic source benchmarks, where each benchmark S has $|S|$ instances. For each model m, source benchmark S, and index $j \in \{1, \ldots, |S|\}$, let $X_S[m, j] \in [0, 1]$ denote m's (possibly non-binary) score on the j-th instance of S. Collecting these gives the per-benchmark source score matrix $X_S \in [0, 1]^{|M| \times |S|}$.

For a held-out evaluation protocol, we partition M into a *calibration* set M_{train} and an *evaluation* set M_{eval}, with calibration models used to fit the selection and predictor, and held-out models used to measure generalization.

Goal A (Performance Prediction). Given a budget $C \in \mathbb{N}$ of source instances, select a set of instance indices $P = \bigsqcup_{S \in \mathcal{S}} P_S$ from the source pool, where $P_S \subseteq \{1, \ldots, |S|\}$ is the set of selected indices for source benchmark S, such that $|P| = \sum_{S \in \mathcal{S}} |P_S| = C$. Then learn a predictor f s.t.

$$\hat{y}_m = f\Big(\{X_S[m, j]\}_{j \in P_S,\ S \in \mathcal{S}}\Big) \approx \bar{y}_m, \tag{1}$$

where the input to f is model m's scores on every selected instance across all source benchmarks. This goal answers the question *what score would a model achieve on the target agentic benchmark?*

Goal B (Pairwise Preference Prediction). Using the same selection process, learn a predictor g that, for any ordered pair $(m, m') \in M \times M$ with $m \neq m'$, outputs a binary label indicating which model is stronger on the target benchmark:

$$\hat{b}_{m,m'} = g\Big(\{X_S[m, j],\ X_S[m', j]\}_{j \in P_S,\ S \in \mathcal{S}}\Big) \in \{0, 1\}, \quad \text{with} \quad \hat{b}_{m,m'} \approx \mathbb{1}[\bar{y}_m > \bar{y}_{m'}]. \tag{2}$$

This goal answers the question *which of two models is stronger on the target agentic benchmark?*

3 PACE: A Proxy for Agentic Capability Evaluation

Building on the problem formulation in § 2.2, we describe PACE, our framework for effectively and efficiently predicting model performances (Goal A) and pairwise model preferences (Goal B) on a target agentic benchmark using non-agentic benchmarks. Figure 2 is a high level overview of PACE.

[1] Appendix B includes citations and descriptions of these benchmarks and capabilities.

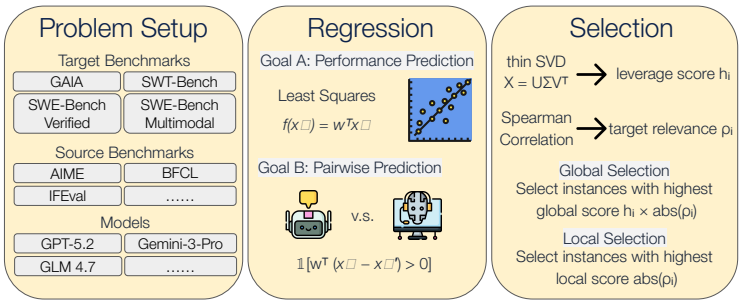

Figure 2: Overview of PACE. From a pool of non-agentic source benchmarks, two complementary filter-based criteria (Local: target relevance; Global: SVD leverage $\times$ relevance) each pick C instances; the selected scores then drive a noise-aware regression that predicts the target agentic benchmark's mean score (Goal A) and pairwise model preferences (Goal B).

PACE consists of two core components: (1) a **regression** that, given selected source instances for each target benchmark, builds a noise-aware predictor for either goals (described in § 3.1); and (2) an **instance selection** method that selects instances via two complementary strategies, **Local** and **Global** selection (§ 3.2). We describe regression first, because the evaluation criterion for selection depends on how the selected instances are used downstream.

3.1 Regression

Our regression predicts a vector $x_m \in \mathbb{R}^C$ collecting model m's scores on the C source instances selected for the target (selection is described in § 3.2). The supervision signal is the target mean $\bar{y}_m = \frac{1}{|T|}\sum_{i=1}^{|T|} y_{m,i}$.

Goal A (Performance Prediction). We fit a linear least-squares regression $f(x_m) = w^\top x_m$ over the training models $m \in M_{\text{train}}$, with the coefficient vector $w \in \mathbb{R}^C$ obtained by minimising $\sum_{m \in M_{\text{train}}} \left(w^\top x_m - \bar{y}_m\right)^2$ (with regularization hyperparameters tuned by held-out evaluation). The prediction for a held-out model $m \in M_{\text{eval}}$ is

$$\hat{y}_m \;=\; w^\top x_m. \tag{3}$$

Goal B (Pairwise Preference Prediction). For each ordered pair $(m, m') \in M_{\text{train}} \times M_{\text{train}}$ with $m \neq m'$, we form the source-score difference $x_m - x_{m'}$ and the label $\mathbb{1}[\bar{y}_m > \bar{y}_{m'}]$. We fit a logistic regressor $g(\Delta x) = w_g^\top \Delta x$ (a structured Bradley-Terry model (Firth, 2005)) on these pair-differences; for a pair (m, m') involving the held-out model m, the predicted logit is

$$\hat{z}_{m,m'} \;=\; w_g^\top (x_m - x_{m'}), \tag{4}$$

and the binary outcome is $\hat{b}_{m,m'} = \mathbb{1}[\hat{z}_{m,m'} > 0]$.

Bootstrapping target instances. Agentic evaluation is expensive, so each target benchmark contains only a few hundred instances, along with a small model pool. Both scarcities make the target mean $\bar{y}_m$ a noisy estimate of model m's true target performance, and treating it as exact lets the regression weights underestimate predictive uncertainty and overfit to one target-instance sample. We therefore draw B bootstrap replicates of each $\bar{y}_m$ by resampling target instances with replacement, yielding $\bar{y}_m^{(b)}$ for $b = 1, \ldots, B$. Replacing $\bar{y}_m$ in the Goal A least-squares and Goal B logistic training objectives defined above with the concatenation of these replicates lets the regression average over the sampling distribution, making the weights estimate less sensitive to target-instance sampling noise.

3.2 Instance Selection

We now turn to selecting the proxy subset P of size C defined in § 2.2. To enable cross-benchmark selection, we stack the per-benchmark score matrices into a single source matrix $X = [X_{S_1} \mid X_{S_2} \mid$

$\cdots \mid X_{S_{|\mathcal{S}|}}] \in [0,1]^{|M|\times|\mathcal{I}|}$ indexed by $\mathcal{I} = \bigsqcup_{S\in\mathcal{S}}\{1,\ldots,|S|\}$, which enumerates all candidate instances across benchmarks (so $|\mathcal{I}| = \sum_{S\in\mathcal{S}}|S|$).

Given this setup, a natural approach is to fit a regularized linear model on X and use the resulting Lasso (Tibshirani, 1996) or Ridge (Hoerl & Kennard, 1970) weights to select the proxy instances. However, in our regime ($|M_{\text{train}}| \ll |\mathcal{I}|$) the joint fit is severely under-determined and overfits (see Appendix E for more details). We thus decouple selection from regression.

Decoupling selection from regression via SVD. We instead score each source instance *independently* using two complementary filter-based signals, both cheap to compute and stable under perturbations of M_{train}. To define them, we perform a thin SVD on X: $X = U\Sigma V^\top$ ($U \in \mathbb{R}^{|M|\times n_c}$, $V \in \mathbb{R}^{|\mathcal{I}|\times n_c}$, where n_c is the SVD rank hyperparameter). The two signals are:

- *Geometric importance*: the leverage score of instance $i \in \mathcal{I}$ in the SVD latent space, defined as $h_i \triangleq \sum_c V_{c,i}^2$, which measures its contribution to the global latent structure of the source pool, a classical criterion for selecting informative columns of a matrix (Mahoney & Drineas, 2009).
- *Target relevance*: $\text{abs}(\rho_i) \triangleq \text{abs}\big(\text{Spearman}(X_{M_{\text{train}},i},\, \bar{y}_{M_{\text{train}}})\big)$, i.e., the rank consistency between the instance score and the target mean across training models. This is a standard filter-based feature-selection criterion (Guyon & Elisseeff, 2003).

These two signals represent a standard tradeoff of *shared geometric prior* versus *task specialization* (Saeys et al., 2007), and PACE draws on both. We *jointly* select a single proxy subset $P = L \cup G$ with $|P| = C$, splitting the budget into two per-strategy sub-budgets $C_L + C_G = C$:

- *Global* subset (G). This strategy jointly considers geometric importance and target relevance, and selects the top-C_G instances by the product of the two signals: $\sigma_i = h_i \times \text{abs}(\rho_i)$. Here V is target-independent and serves as a *prior* over the global latent structure of the source pool: leverage $h_i = \sum_c V_{c,i}^2$ identifies information-rich instances, with target relevance applied as a secondary filter. The held-out model $m^\star \in M_{\text{eval}}$ is not in the SVD decomposition, so we obtain its latent-space coordinates separately. We project its score row $X_{m^\star,:}$ onto V via the pseudoinverse V^+. This places $m^\star$ in the same n_c-dimensional latent space as the training models, allowing the regression to operate uniformly across all models.
- *Local* subset (L). This strategy selects the top-C_L instances solely by target relevance $\text{abs}(\rho_i)$ (without using any geometric signal), and then *recomputes* the SVD on the $|M| \times |L|$ submatrix X_L to obtain a local basis V_{loc}. As in Global selection, the held-out model $m^\star$ is projected into this basis via the pseudoinverse V_{loc}^+, applied to its restricted score row $X_{m^\star,L}$ over the selected C_L instances. The resulting embedding lives in the *local* SVD space adapted to the selected subset rather than the global pool.

The two subsets are complementary: Global alone may select high-leverage instances that are irrelevant to the target, whereas Local alone ignores any geometric structure outside the selected subset. PACE therefore uses both, combining them at prediction time rather than at selection time.

When the two top-ranked sets overlap, $|L \cup G|$ falls short of C; we greedily extend each side past columns already in $L \cup G$ (in proportion to the $C_L : C_G$ split) until $|L \cup G| = C$, so only C unique source instances are evaluated per new model. PACE's final output is the ensemble $\hat{y} = \lambda \cdot \hat{y}_L + (1-\lambda)\cdot \hat{y}_G$, with both hyperparameters the ensemble weight λ and the budget split (C_L, C_G) optimized through held-out validation, letting the data rather than prior assumptions determine these ratios.

4 Experiments and Results

4.1 Experimental Setup

Target Agentic Benchmarks. We evaluate PACE on all 4 agentic benchmarks from Table 1. These benchmarks vary in terms of tasks, required model abilities, and model performances; together they provide broad coverage of agentic tasks, including browser-based question answering, repository-level code generation, multimodal software engineering, and test understanding and construction.

- **GAIA** (Mialon et al., 2024) is a benchmark for general-purpose AI assistants, consisting of real-world questions that require reasoning, tool use, web browsing, and, in some cases, multimodal understanding. It evaluates whether agents can decompose underspecified information-seeking tasks, gather evidence from external sources, and produce concise answers.
- **SWE-Bench Verified** (Jimenez et al., 2024) evaluates repository-level software engineering agents on real GitHub issues. Given a codebase and an issue description, an agent must localize the relevant code, implement a patch, and pass the corresponding tests, making it a benchmark for practical bug fixing and code modification.
- **SWE-Bench Multimodal** (Yang et al., 2025) extends software-engineering evaluation to issues that contain visual information, such as screenshots, mockups, diagrams, or visually presented error messages. It tests whether agents can combine visual understanding with repository-level code reasoning to resolve realistic software issues.
- **SWT-Bench** (Mündler et al., 2024) evaluates agents on software test generation for real-world bug fixes. Given the original codebase and a user-reported issue, the agent must generate tests that expose the bug by failing before the fix and passing after the fix, thereby measuring its ability to understand intended behavior and construct effective validation tests.

All agentic benchmark results are obtained using the OpenHands Index (OpenHands Team, 2025), which evaluates models with the OpenHands Software Agent SDK (Wang et al., 2025) under a unified agent harness, enabling fair cross-model comparisons.

Source Non-Agentic Benchmarks. The candidate source pool consists of all 19 non-agentic benchmarks from Table 1. These benchmarks collectively cover all 11 capabilities identified in § 2.1. We evaluate non-agentic benchmarks using `lm-evaluation-harness` (Gao et al., 2024); for benchmarks not yet supported by this package, we use each benchmark's official evaluation code.

Models. We evaluate across 14 models spanning proprietary and open models, including GPT 5.2 (OpenAI, 2025a), GPT 5.2 Codex (OpenAI, 2025b), Gemini 3 Pro Preview (Gemini Team, 2025), Gemini 3 Flash Preview (Gemini Team, 2025), Claude Opus 4.5 (Anthropic, 2025a) and 4.6 (Anthropic, 2026), Claude Sonnet 4.5 (Anthropic, 2025b), DeepSeek V3.2 (Liu et al., 2025), GLM 4.7 (Z.ai, 2025), Kimi K2 Team et al. (2026b) and K2.5 Team et al. (2026a), MiniMax M2.1 (MiniMax, 2025) and M2.5 (MiniMax, 2026), and Qwen3 Coder 480B A35B (Team, 2025).

Evaluation Protocol. We evaluate PACE under a strict **LOOCV (leave-one-out cross-validation)** protocol. In each fold, one of the 14 models is held out as m^*, while the remaining models are used for source-instance selection and regression. We then aggregate the held-out predictions across all 14 folds. This protocol answers the central question: *can* PACE *generalize to unseen models?*

4.2 Main Results

Table 2 reports the predictive power of PACE with $C = 100$ proxy instances across all four agentic benchmarks for both Goal A and Goal B.

	Goal A			Goal B
Target	**MAE**	**Spearman Corr.**	**Pearson Corr.**	**Accuracy**
GAIA	5.77%	0.79	0.78	83.33%
SWE-bench Verified	2.09%	0.67	0.61	78.54%
SWE-bench Multimodal	2.23%	0.89	0.80	85.39%
SWT-bench	5.12%	0.89	0.78	90.11%
Average	**3.80%**	**0.81**	**0.74**	**84.37%**

Table 2: Strict LOOCV results on Goal A (absolute score prediction; MAE, Spearman, and Pearson correlation) and Goal B (pairwise preference prediction; accuracy).

Goal A (Performance Prediction). PACE achieves an average MAE of 3.80% and a Spearman correlation of 0.807 on Goal A, showing that only 100 non-agentic source instances are sufficient to accurately predict absolute scores on agentic benchmarks. Although GAIA and SWT-bench exhibit slightly larger absolute errors, they still show strong ranking performance (Spearman ≥ 0.79), suggesting that under LOOCV the dominant signal is model-capability ordering rather than precise numerical calibration.

Goal B (pairwise preference prediction). Reusing the same selected instances under a pairwise logistic, PACE achieves an average LOOCV pair-accuracy of 84.4%, far above the random 50% baseline.

4.3 Trade Off of Performance v.s. Cost of PACE-BENCH

Figure 1 plots the cost-versus-quality tradeoff of PACE (blue) against a random target-sampling baseline (red), averaged across the four agentic targets. The left plot reports LOOCV MAE (lower is better), the middle one reports LOOCV Spearman correlation (higher is better), and the right one reports pairwise model-ranking accuracy (higher is better). At every budget below saturation, PACE dominates random target-sampling on all three metrics, matching PACE's quality with the random target-sampling baseline, but costing only roughly $\frac{1}{100}$ the cost of the baseline, demonstrating the effectiveness and efficiency of PACE. We report the full budget sweep of PACE as C ranges from 25 to 500 in Appendix D; quality broadly improves with the budget and then saturates, with $C = 100$ sitting at a practical sweet spot.

5 Analysis and Discussion

5.1 What Does Learned Allocation Select?

Figure 3: Total selected instances from source benchmarks covering each ability, for $C = 100$ across the four agentic targets (abbreviations follow Table 1).

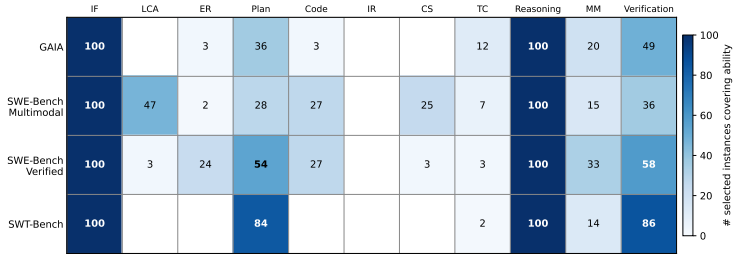

Figure 3 shows the ability distribution (ability categorization shown in Table 1) of the $C = 100$ selected source instances of PACE-BENCH, revealing what each agentic target requires from its proxy. The distribution for each source benchmark could be found in Appendix C.

Instruction Following and Reasoning are saturated at 100 selected instances for all four targets, since every source benchmark in our pool covers these abilities. The remaining capabilities vary substantially across targets, showing what each agentic benchmark uniquely requires.

GAIA: Instruction Following and Verification and Test. GAIA's allocation is overwhelmingly drawn from IF-focused benchmarks IFEval and PlanBench, along with a focus on Verification and Test. This matches GAIA's nature as browser-based question answering: each task specifies multi-clause answer-format and browsing constraints, so success hinges on rigorously following the prompt and self-checking that the produced answer satisfies every clause.

SWE-Bench Verified: Planning, Verification and Test, Code Generation, and Error Recovery. These abilities reveal the core requirement of SWE-Bench Verified: the model must plan a code edit, write the patch, run the hidden test suite, and recover when tests fail.

SWE-Bench Multimodal: long-context aggregation. SWE-Bench Multimodal is dominated by Long-Context Aggregation, as its tasks require integrating information across long issues, repository code, and screenshots. Notably, Multimodal allocation is comparatively modest, because many

SWE-bench Multimodal instances can be solved primarily from the textual issue and code with the visual contents merely as a supporting role.

SWT-Bench: Verification and Test and Planning. SWT-Bench is unusually concentrated, peaking on Verification and Planning. The benchmark asks the model to author tests that exercise specific bug-triggering paths, which requires planning a multi-step test sequence and reasoning carefully about what each assertion checks, precisely the Verification and Test and Planning abilities.

Overall, Figure 3 suggests that the selection process could capture both a shared capability requirement of agentic tasks and benchmark-specific capability signatures.

5.2 Bootstrap

To isolate the effect of the bootstrap pooling described in § 3.1, we perform an ablation by removing the bootstrap step, holding selection and regression fixed. Table 3 reports the resulting LOOCV scores alongside their with-bootstrap counterparts from Table 2.

Table 3: No-bootstrap LOOCV results compared against the with-bootstrap LOOCV results in Table 2. Δ MAE and Δ Spearman are computed as with-bootstrap minus no-bootstrap.

Target	No-bootstrap MAE	No-bootstrap Sp.	Δ MAE	Δ Sp.
GAIA	6.45%	0.71	-0.68%	$+0.08$
SWE-bench Verified	2.75%	0.39	-0.66%	$+0.28$
SWE-bench Multimodal	3.40%	0.69	-1.17%	$+0.20$
SWT-bench	5.68%	0.82	-0.56%	$+0.07$
Average	**4.57%**	**0.66**	$\mathbf{-0.77\%}$	$\mathbf{+0.15}$

Bootstrap helps on every target: average MAE improves by 0.77% and average Spearman by $+0.15$, with no target regressing on either metric. These results justify keeping the bootstrap as the default configuration of PACE.

6 Conclusion

We presented PACE, a framework for predicting agentic-benchmark performance from a small, automatically selected set of non-agentic source instances. PACE combines noise-aware bootstrap regression with two complementary filter-based selection signals: SVD leverage as a target-independent geometric prior, and rank correlation as a target-specific relevance score. These signals are ensembled with a learned per-target weight, allowing PACE to identify compact, informative proxy subsets for each agentic benchmark.

Across four agentic targets and 14 models, PACE achieves 3.80% MAE, 0.81 Spearman correlation, and around 85% pairwise accuracy under leave-one-out cross-validation, while costing roughly $100\times$ less than a random target-sampling baseline of matched quality. Beyond prediction accuracy, the selected source instances provide interpretable per-target capability profiles, revealing each agentic benchmark's distinctive ability mix without requiring human-supplied annotations.

These results suggest that PACE can make agentic evaluation substantially more practical during model development. Developers can use it to cheaply rank candidate models before running full agentic evaluations, monitor training checkpoints or hyperparameter runs with much denser feedback, and gate expensive full-harness evaluations by first screening whether a model is likely to be competitive. In this way, PACE reduces agentic evaluation from hours of harness setup and hundreds of API calls per model to minutes of static-benchmark evaluation at roughly $\frac{1}{100}$ of the cost.

The main limitation is that these use cases depend on the calibration models being representative of future models. When a new model falls outside the calibration distribution, or reflects a substantially different architecture or training paradigm, proxy error may increase. In practice, this means the calibration set should be refreshed periodically as model distributions shift. We hope PACE makes agentic evaluation routinely affordable for model development and serves as a foundation for future work on automatic, scalable benchmarking of agentic capabilities.

References

Anthropic. Introducing claude opus 4.5, 2025a. URL https://www.anthropic.com/news/claude-opus-4-5.

Anthropic. Introducing claude sonnet 4.5, 2025b. URL https://www.anthropic.com/news/claude-sonnet-4-5.

Anthropic. Introducing claude opus 4.6, 2026. URL https://www.anthropic.com/news/claude-opus-4-6.

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

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the trenches on reproducible evaluation of language models, 2024. URL https://arxiv.org/abs/2405.14782.

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

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Zhiyu Fan, Kirill Vasilevski, Dayi Lin, Boyuan Chen, Yihao Chen, Zhiqing Zhong, Jie M. Zhang, Pinjia He, and Ahmed E. Hassan. Swe-effi: Re-evaluating software ai agent system effectiveness under resource constraints, 2025. URL https://arxiv.org/abs/2509.09853.

David Firth. Bradley-terry models in r. *Journal of Statistical software*, 12:1–12, 2005.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gemini Team. Gemini 3: A new era of intelligence with gemini 3, 2025. URL https://blog.google/products-and-platforms/products/gemini/gemini-3/#gemini-3.

Adhiraj Ghosh, Sebastian Dziadzio, Ameya Prabhu, Vishaal Udandarao, Samuel Albanie, and Matthias Bethge. ONEBench to test them all: Sample-level benchmarking over open-ended capabilities. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32445–32481, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1560. URL https://aclanthology.org/2025.acl-long.1560/.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021a.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021b.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Alex Kipnis, Konstantinos Voudouris, Luca M. Schulze Buschoff, and Eric Schulz. metabench - a sparse benchmark of reasoning and knowledge in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=4T33izzFpK.

Harsha Kokel, Michael Katz, Kavitha Srinivas, and Shirin Sohrabi. Acpbench: Reasoning about action, change, and planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26559–26568, 2025.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue WANG, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Andrew Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=iO4LZibEqW. Featured Certification, Expert Certification, Outstanding Certification.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Junpeng Liu, Yifan Song, Bill Yuchen Lin, Wai Lam, Graham Neubig, Yuanzhi Li, and Xiang Yue. Visualwebbench: How far have multimodal llms evolved in web page understanding and grounding? *arXiv preprint arXiv:2404.05955*, 2024.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. *arXiv preprint arXiv:2306.03091*, 2023.

Michael W Mahoney and Petros Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 106(3):697–702, 2009.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=fibxvahvs3.

MiniMax. Minimax m2.1: Significantly enhanced multi-language programming, built for real-world complex tasks, 2025. URL https://www.minimax.io/news/minimax-m21.

MiniMax. Minimax m2.5: Built for real-world productivity., 2026. URL https://www.minimax.io/news/minimax-m25.

Niels Mündler, Mark Niklas Mueller, Jingxuan He, and Martin Vechev. SWT-bench: Testing and validating real-world bug-fixes with code agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=9Y8zUO11EQ.

OpenAI. Introducing gpt-5.2, 2025a. URL https://openai.com/index/introducing-gpt-5-2/.

OpenAI. Introducing gpt-5.2-codex, 2025b. URL https://openai.com/index/introducing-gpt-5-2-codex/.

OpenHands Team. Openhands index: A comprehensive leaderboard for ai coding agents. https://index.openhands.dev, 2025.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=2GmDdhBdDk.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. Efficient benchmarking (of language models). In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 2519–2536, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.139. URL https://aclanthology.org/2024.naacl-long.139/.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jiarui Qin, Yunjia Xi, Junjie Huang, Renting Rui, Di Yin, Weiwen Liu, Yong Yu, Weinan Zhang, and Xing Sun. Aptbench: Benchmarking agentic potential of base llms during pre-training, 2025. URL https://arxiv.org/abs/2510.24397.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. Infobench: Evaluating instruction following ability in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13025–13048, 2024.

Ashwin Ramaswamy, Nestor Demeure, and Ermal Rrapaj. Model consistency as a cheap yet predictive proxy for LLM elo scores. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 30167–30175, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1534. URL https://aclanthology.org/2025.emnlp-main.1534/.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

Yueqi Song, Tianyue Ou, Yibo Kong, Zecheng Li, Graham Neubig, and Xiang Yue. Visualpuzzles: Decoupling multimodal reasoning evaluation from domain knowledge. *arXiv preprint arXiv:2504.10342*, 2025.

Lucas Spangher, Tianle Li, William F. Arnold, Nick Masiewicki, Xerxes Dotiwalla, Rama Kumar Pasumarthi, Peter Grabowski, Eugene Ie, and Daniel Gruhl. Chatbot arena estimate: towards a generalized performance benchmark for LLM capabilities. In Weizhu Chen, Yi Yang, Mohammad Kachuee, and Xue-Yong Fu (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: Industry Track)*, pp. 1016–1025, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-194-0. doi: 10.18653/v1/2025.naacl-industry.77. URL https://aclanthology.org/2025.naacl-industry.77/.

Theodore Sumers, Shunyu Yao, Karthik R Narasimhan, and Thomas L Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2023.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, S. H. Cai, Yuan Cao, Y. Charles, H. S. Che, Cheng Chen, Guanduo Chen, Huarong Chen, Jia Chen, Jiahao Chen, Jianlong Chen, Jun Chen, Kefan Chen, Liang Chen, Ruijue Chen, Xinhao Chen, Yanru Chen, Yanxu Chen, Yicun Chen, Yimin Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Ziwei Chen, Dazhi Cheng, Minghan Chu, Jialei Cui, Jiaqi Deng, Muxi Diao, Hao Ding, Mengfan Dong, Mengnan Dong, Yuxin Dong, Yuhao Dong, Angang Du, Chenzhuang Du, Dikang Du, Lingxiao Du, Yulun Du, Yu Fan, Shengjun Fang, Qiulin Feng, Yichen Feng, Garimugai Fu, Kelin Fu, Hongcheng Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Chengyang Gong, Xiaochen Gong, Zhuoma Gongque, Qizheng Gu, Xinran Gu, Yicheng Gu, Longyu Guan, Yuanying Guo, Xiaoru Hao, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Jiaxi Hu, Yangyang Hu, Zhenxing Hu, Ke Huang, Ruiyuan Huang, Weixiao Huang, Zhiqi Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yu Jing, Guokun Lai, Aidi Li, C. Li, Cheng Li, Fang Li, Guanghe Li, Guanyu Li, Haitao Li, Haoyang Li, Jia Li, Jingwei Li, Junxiong Li, Lincan Li, Mo Li, Weihong Li, Wentao Li, Xinhang Li, Xinhao Li, Yang Li, Yanhao Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zheming Li, Weilong Liao, Jiawei Lin, Xiaohan Lin, Zhishan Lin, Zichao Lin, Cheng Liu, Chenyu Liu, Hongzhang Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Tianyu Liu, Weizhou Liu, Xiangyan Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yuanxin Liu, Yue Liu, Zhengying Liu, Zhongnuo Liu, Enzhe Lu, Haoyu Lu, Zhiyuan Lu, Junyu Luo, Tongxu Luo, Yashuo Luo, Long Ma, Yingwei Ma, Shaoguang Mao, Yuan Mei, Xin Men, Fanqing Meng, Zhiyong Meng, Yibo Miao, Minqing Ni, Kun Ouyang, Siyuan Pan, Bo Pang, Yuchao Qian, Ruoyu Qin, Zeyu Qin, Jiezhong Qiu, Bowen Qu, Zeyu Shang, Youbo Shao, Tianxiao Shen, Zhennan Shen, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Feifan Song, Pengwei Song, Tianhui Song, Xiaoxi Song, Hongjin Su, Jianlin Su, Zhaochen Su, Lin Sui, Jinsong Sun, Junyao Sun, Tongyu Sun, Flood Sung, Yunpeng Tai, Chuning Tang, Heyi Tang, Xiaojuan Tang, Zhengyang Tang, Jiawen Tao, Shiyuan Teng, Chaoran Tian, Pengfei Tian, Ao Wang, Bowen Wang, Chensi Wang, Chuang Wang, Congcong Wang, Dingkun Wang, Dinglu Wang, Dongliang Wang, Feng Wang, Hailong Wang, Haiming Wang, Hengzhi Wang, Huaqing Wang, Hui Wang, Jiahao Wang, Jinhong Wang, Jiuzheng Wang, Kaixin Wang, Linian Wang, Qibin Wang, Shengjie Wang, Shuyi Wang, Si Wang, Wei Wang, Xiaochen Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yipu Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhexu Wang, Zihan Wang, Zizhe Wang, Chu Wei, Ming Wei, Chuan Wen, Zichen Wen, Chengjie Wu, Haoning Wu, Junyan Wu, Rucong Wu, Wenhao Wu, Yuefeng Wu, Yuhao Wu, Yuxin Wu, Zijian Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Yuchong Xie, Yifei Xin, Bowei Xing, Boyu Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinbo Xu, Xinran Xu, Yangchuan Xu, Yichang Xu, Yuemeng Xu, Zelai Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Guangyao Yang, Hao Yang, Junwei Yang, Kai Yang, Ningyuan Yang, Ruihan Yang, Xiaofei Yang, Xinlong Yang, Ying Yang, Yi Yang, Yi Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Dan Ye, Wenjie Ye, Zhuorui Ye, Bohong Yin, Chengzhen Yu, Longhui Yu, Tao Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Xiaokun Yuan, Yang Yue, Weihao Zeng, Dunyuan Zha, Haobing Zhan, Dehao Zhang, Hao Zhang, Jin Zhang, Puqi Zhang, Qiao Zhang, Rui Zhang, Xiaobin Zhang, Y. Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yushun Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Chenguang Zhao, Feifan Zhao, Jinxiang Zhao, Shuai Zhao, Xiangyu Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Junfeng Zhong, Longguang Zhong, Weiming Zhong, M. Zhou, Runjie Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yuxuan Zhu, Zhen Zhu, Jingze Zhuang, Weiyu Zhuang, Ying Zou, and Xinxing Zu. Kimi k2.5: Visual agentic intelligence, 2026a. URL https://arxiv.org/abs/2602.02276.

Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, Guanduo Chen, Haiting Chen, Huarong Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Chenxiao Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Yuyao Ge, Shangyi Geng, Qizheng Gu, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Yunjia He, Chao Hong, Hao Hu, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yang Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Haoyu Lu, Lijun Lu, Yashuo Luo, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Zeyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Lin Sui, Xinjie Sun, Flood Sung, Yunpeng Tai, Heyi Tang, Jiawen Tao, Qifeng Teng, Chaoran Tian, Chensi Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Si Wang, Xinyuan Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Haoning Wu, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Jin Xie, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Jing Xu, Jing Xu, Junjie Yan, Yuzi Yan, Hao Yang, Xiaofei Yang, Yi Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Siyu Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yadong Zhang, Yangkun Zhang, Yichi Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Zijia Zhao, Huabin Zheng, Shaojie Zheng, Longguang Zhong, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Jinguo Zhu, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2026b. URL https://arxiv.org/abs/2507.20534.

Qwen Team. Qwen3-coder: Agentic coding in the world, 2025. URL https://qwen.ai/blog?id=qwen3-coder.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Yinxu Pan, Yesai Wu, Hui Haotian, Liu Weichuan, Zhiyuan Liu, et al. Debugbench: Evaluating debugging capability of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4173–4198, 2024.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 01 1996. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987, 2023.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024a. ISSN 2095-2236. doi: 10.1007/s11704-024-40231-1. URL http://dx.doi.org/10.1007/s11704-024-40231-1.

Xingyao Wang, Simon Rosenberg, Juan Michelini, Calvin Smith, Hoang Tran, Engel Nyst, Rohit Malhotra, Xuhui Zhou, Valerie Chen, Robert Brennan, and Graham Neubig. The openhands

software agent sdk: A composable and extensible foundation for production agents, 2025. URL https://arxiv.org/abs/2511.03690.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024b.

Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. Lifbench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16445–16468, 2025.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

John Yang, Carlos E Jimenez, Alex L Zhang, Kilian Lieret, Joyce Yang, Xindi Wu, Ori Press, Niklas Muennighoff, Gabriel Synnaeve, Karthik R Narasimhan, Diyi Yang, Sida Wang, and Ofir Press. Swe-bench multimodal: Do ai systems generalize to visual software domains? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=riTiq3i21b.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Z.ai. Glm-4.7: Advancing the coding capability, 2025. URL https://z.ai/blog/glm-4.7.

Guanhua Zhang, Florian E. Dorner, and Moritz Hardt. How benchmark prediction from fewer data misses the mark. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*, 2025. URL https://openreview.net/forum?id=3pUtKEctwR.

Qiyuan Zhang, Fuyuan Lyu, Xue Liu, and Chen Ma. Collaborative performance prediction for large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2576–2596, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.150. URL https://aclanthology.org/2024.emnlp-main.150/.

Taolin Zhang, Hang Guo, Wang Lu, Tao Dai, Shu-Tao Xia, and Jindong Wang. Sparseeval: Efficient evaluation of large language models by sparse optimization. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CZAzAedGSV.

Yifan Zhang and Team Math-AI. American invitational mathematics examination (aime) 2025, 2025.

Hongli Zhou, Hui Huang, Ziqing Zhao, Lvyuan Han, Huicheng Wang, Kehai Chen, Muyun Yang, Wei Bao, Jian Dong, Bing Xu, Conghui Zhu, Hailong Cao, and Tiejun Zhao. Lost in benchmarks? rethinking large language model benchmarking with item response theory, 2026. URL https://arxiv.org/abs/2505.15055.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Related Work

Although agentic benchmarks provide more direct measurements of agent behaviors, their multi-turn and execution-based nature introduces major practical bottlenecks. To reduce the significant cost of evaluation, a growing line of work studies how to approximate benchmark scores using fewer examples. Methods such as tinyBenchmarks (Polo et al., 2024), PSN-IRT (Zhou et al., 2026), metabench (Kipnis et al., 2025), and SparseEval (Zhang et al., 2026) identify highly discriminative subsets from within a target benchmark, using item response theory, learned task representations, sparse optimization (Perlitz et al., 2024).

Other approaches such as APTBench (Qin et al., 2025) convert agentic trajectories into static multiple-choice questions, and SWE-bench Verified (Chowdhury et al., 2024) reduces the original benchmark to a human-validated subset of 500 instances for more reliable evaluation. These methods aim to preserve signal about the original benchmark while lowering cost. All of these methods operate within a single benchmark distribution, while the problem of reliably predicting performance on an agentic benchmark from an external pool of heterogeneous and inexpensive evaluation benchmarks remains largely unaddressed (Zhang et al., 2025).

B Benchmark and Capabilities

Table 1 provides a representative list of benchmarks from two settings: standard LLM-based static benchmarks ACPBench (Kokel et al., 2025), AIME (Zhang & Math-AI, 2025), BEIR (Thakur et al., 2021), BFCL (Patil et al., 2025), DebugBench (Tian et al., 2024), GPQA (Rein et al., 2024), HumanEval (Chen et al., 2021), IFEval (Zhou et al., 2023), InFoBench (Qin et al., 2024), LIFBench (Wu et al., 2025), LiveCodeBench (Jain et al., 2024), LogiQA (Liu et al., 2020), MBPP (Austin et al., 2021), MMLU (Hendrycks et al., 2020), MMMU (Yue et al., 2024), PlanBench (Valmeekam et al., 2023), RepoBench (Liu et al., 2023), VisualPuzzles (Song et al., 2025), and VisualWebBench (Liu et al., 2024); and agentic benchmarks including GAIA (Mialon et al., 2024), SWE-Bench Verified (Jimenez et al., 2024), SWE-Bench Multimodal (Yang et al., 2025), SWT-Bench (Mündler et al., 2024).

We organize model capabilities into 11 categories: instruction following (adhering to explicit constraints), long context aggregation (aggregation over extended inputs), error recovery (correcting intermediate mistakes), planning (sequencing multi-step actions), code generation (writing executable programs), information retrieval (locating relevant content), code search (navigating and understanding codebases), tool calling (invoking structured APIs), reasoning (logical and mathematical inference), multimodal understanding (interpreting visual inputs), verification and test (checking correctness of outputs and reasoning about test cases).

C Per-source-benchmark allocation

Figure 4 shows the same $C = 100$ selected instances analyzed in § 5.1, but disaggregated by the originating source benchmark rather than aggregated over the abilities each benchmark covers. This view exposes which specific source benchmarks PACE draws from for each target, and complements the capability-level discussion in the main text.

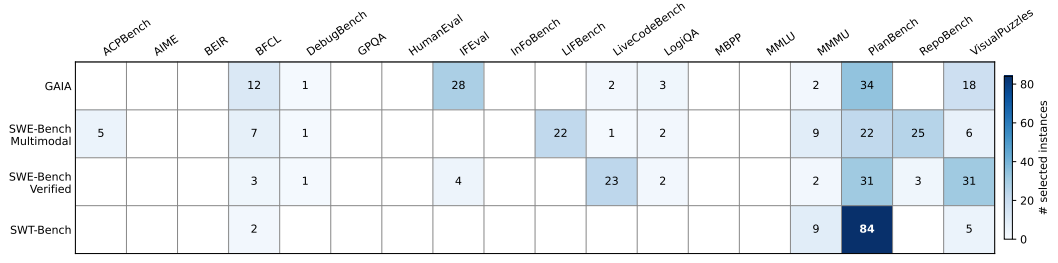

Figure 4: Number of source instances selected per (target, source benchmark) pair at $C = 100$. Cells are deduplicated across the Local and Global selection lists, so each row sums to exactly 100.

Shared predictors across targets. A small set of source benchmarks is consistently selected across all four targets. PlanBench is the single largest contributor to every target (34 for GAIA, 31 for SWE-bench Verified, 22 for SWE-bench Multimodal, 84 for SWT-bench), reflecting that planning–verification chains generalize across heterogeneous agentic surfaces. VisualPuzzles (18/31/6/5), BFCL (12/3/7/2), and MMMU (2/2/9/9) also appear for every target, supplying multimodal-reasoning and tool-use signals that discriminate frontier models even when the target task itself is unimodal.

Target-specific benchmark allocation. Beyond the shared substrate, each target draws on a distinctive set of source benchmarks that match its surface form:

- **SWE-bench Verified** concentrates on LiveCodeBench (23) and VisualPuzzles (31), aligning with patch-generation tasks that require both code synthesis and structured reasoning over long programs.
- **SWE-bench Multimodal** relies heavily on RepoBench (25) and LIFBench (22), reflecting its long-context navigation across code, screenshots, and issues—both selected source benchmarks emphasize long-context aggregation over codebases.
- **GAIA** is dominated by IFEval (28) and PlanBench (34), matching its browser-based question-answering style that combines strict instruction following with multi-step planning.
- **SWT-bench** is the most concentrated target: 84 of its 100 selected instances come from PlanBench, consistent with that benchmark's emphasis on planning multi-step test generation and validating outputs against specifications.

D Budget Sweep

Table 4: Goal A / Goal B LOOCV performance as a function of the source-instance budget C, averaged over the 4 agentic targets. Best value per column in bold.

C	Goal A MAE	Goal A Sp.	Goal B Acc.
25	4.02%	0.832	83.98%
50	4.06%	0.794	83.00%
100	3.80%	0.807	84.37%
200	3.63%	0.832	86.76%
300	3.51%	0.838	86.76%
400	**3.30%**	**0.868**	86.20%
500	3.44%	0.855	**89.27%**

Table 4 reports LOOCV performance as the source-instance budget C is swept from 25 to 500, averaged over the 4 agentic targets. Goal A scales gracefully: MAE broadly improves as the budget grows, reaching its minimum at $C = 400$ (3.30%, with Spearman 0.868) before bending back at $C = 500$ (3.44%), suggesting that the small calibration set starts to overfit when too many regressors are admitted. Goal B accuracy, by contrast, continues climbing all the way to $C = 500$ (89.27%), reflecting that pairwise ranking benefits from additional discriminating instances longer than absolute prediction does.

Even small budgets are highly informative: at $C = 25$ PACE already achieves 4.02% MAE, 0.832 Spearman, and 83.98% pair accuracy—within $\sim 0.7\%$ MAE of the $C = 400$ optimum and within $\sim 5\%$ pair accuracy of the $C = 500$ optimum, while running at a fraction of the eval cost (Figure 1). $C = 100$, the headline budget used elsewhere in the paper, sits at a practical sweet spot: it remains competitive with the larger budgets while keeping per-model evaluation cost low.

E Lasso and Ridge Baseline

We compare PACE to two baselines that perform selection and prediction simultaneously: (1) an ℓ_1-penalized **Lasso** (Tibshirani, 1996) whose non-zero columns form the selected subset, and (2) an ℓ_2-penalized **Ridge** (Hoerl & Kennard, 1970) whose top-C columns by $|w_i|$ form the subset (with Ridge then refit on those C columns). Per LOOCV fold, each baseline is fit on the $|M_{\text{train}}|$

training models; the resulting selection is then reused as the input to a separate pair-logistic for Goal B (mirroring PACE's pinned-selection protocol). To give each baseline its fairest chance, we sweep the regularization strength $\alpha \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ *per target* and report each baseline at the $\alpha^\star$ that minimises its own LOOCV MAE.

Table 5: Comparison of PACE against two baselines, Lasso (L1) and Ridge (L2) with top-C ($C = 100$) selection by $|w_i|$ on Goal A (absolute score; MAE / Spearman / Pearson) and Goal B (pairwise accuracy). For each baseline, the regularization strength $\alpha^\star$ is selected per target from the grid $\alpha \in \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ to minimise LOOCV MAE; the column $\alpha^\star$ reports the chosen value. Left: in-sample fit on all 14 models; right: strict LOOCV.

			In-sample fit				**LOOCV**			
			Goal A			**Goal B**	**Goal A**			**Goal B**
Target	**Method**	$\alpha^\star$	**MAE**	**Sp.**	**Pear.**	**Acc.**	**MAE**	**Sp.**	**Pear.**	**Acc.**
GAIA	PACE	—	4.66%	0.93	0.91	92.22%	5.77%	0.79	0.78	83.33%
	Lasso	10^{-4}	0.02%	1.00	1.00	99.45%	8.62%	0.38	0.40	81.87%
	Ridge	10^{-1}	1.07%	0.98	0.99	95.60%	7.10%	0.79	0.75	86.26%
SWE-bench Verified	PACE	—	1.86%	0.57	0.66	92.13%	2.09%	0.67	0.61	78.54%
	Lasso	10^{-5}	0.00%	1.00	1.00	98.90%	2.82%	0.15	0.16	71.98%
	Ridge	10^{-1}	0.19%	1.00	1.00	97.80%	2.71%	0.66	0.39	70.88%
SWE-bench Multimodal	PACE	—	2.03%	0.87	0.88	95.51%	2.23%	0.89	0.80	85.39%
	Lasso	10^{-2}	2.00%	0.97	0.92	91.21%	3.53%	0.60	0.57	79.67%
	Ridge	10^{-4}	1.00%	0.95	0.94	92.86%	2.79%	0.80	0.83	82.42%
SWT-bench	PACE	—	4.81%	0.86	0.85	87.91%	5.12%	0.89	0.78	90.11%
	Lasso	10^{-5}	0.00%	1.00	1.00	100.00%	7.77%	0.70	0.61	84.62%
	Ridge	10^{-1}	1.04%	0.94	0.99	93.96%	8.03%	0.78	0.64	82.97%
Average	PACE	—	**3.34%**	**0.81**	**0.82**	**91.94%**	**3.80%**	**0.81**	**0.74**	**84.37%**
	Lasso	—	**0.51%**	**0.99**	**0.98**	**97.39%**	**5.69%**	**0.46**	**0.43**	**79.53%**
	Ridge	—	**0.82%**	**0.97**	**0.98**	**95.05%**	**5.16%**	**0.76**	**0.65**	**80.63%**

Table 5 compares PACE and the two baselines under identical in-sample fit and LOOCV across the agentic targets. We highlight three findings.

Both baselines overfit but PACE does not. Even with α tuned per target, both baselines fit near-perfectly in-sample but degrade sharply under LOOCV. PACE, by contrast, has an in-sample-to-LOOCV gap of only 0.46 pp on MAE and ≈ 0 on Spearman. The pair preference accuracies tell the same story. The difference in these gaps is direct evidence that Lasso and Ridge overfits the small set of models, while PACE's decoupled filter scoring generalizes across models.

The conclusion is robust to α. Table 6 reports the average LOOCV MAE and Spearman of each baseline at 5 different values of α. PACE outperforms both baselines at every α tested.

Table 6: Average LOOCV performance (across the 4 agentic targets) at each α on the sweep grid. PACE outperforms both baselines at *every* α tested, with the best-α row reproduced from Table 5 for reference.

	Lasso		**Ridge (top-C)**	
α	MAE	Sp.	MAE	Sp.
10^{-5}	5.83%	0.44	8.83%	0.72
10^{-4}	5.84%	0.41	7.30%	0.72
10^{-3}	5.85%	0.40	6.11%	0.73
10^{-2}	6.63%	0.28	5.88%	0.71
10^{-1}	7.49%	0.00	5.18%	0.75
best $\alpha^\star$ (per target)	**5.69%**	**0.46**	**5.16%**	**0.76**

F Limitations

Proxy gaming. Because PACE selects a fixed, public set of proxy instances, a model developer who knows the proxy set could optimize specifically for those instances-achieving high proxy scores without improving genuine agentic capability. This risk is analogous to benchmark contamination and grows as the proxy set becomes widely known. Mitigations include periodically refreshing the proxy set, keeping it private until evaluation, or sampling a fresh subset at evaluation time.

Small calibration set. Our experiments use 14 frontier models as the calibration set, which is small relative to the $C = 100$ feature dimensionality. Ridge regularization partially addresses this, but the learned regression weights may not be reliable for individual instances; they are better interpreted as aggregate signals. As more models are evaluated on the source benchmarks, the calibration set will grow and selection quality is expected to improve.

Coverage of agentic benchmarks. We evaluate on four agentic benchmarks sharing a common agent framework (OpenHands). Whether PACE generalizes to agentic benchmarks with different scaffolds, tool sets, or evaluation protocols (e.g., browser-based or embodied agents) remains to be studied.

Static source pool. The 19 source benchmarks were chosen to cover a broad range of capabilities, but the proxy set is only as good as the source pool. If a target agentic benchmark requires capabilities not measured by any source benchmark, PACE cannot recover the missing signal regardless of which instances it selects.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction claim that PACE predicts agentic-benchmark performance from a small set of non-agentic instances at substantially lower cost; these claims are supported by the LOOCV results in § 4.2 (Table 2) and the cost–quality comparison in Figure 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: **??** contains a Limitations paragraph explicitly noting that predictions rely on the calibration models being representative of future models and that proxy error may grow under distribution shift.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include formal theorems; the regression and selection procedures are described as concrete algorithms in § 3.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: § 3 fully specifies the regression and selection algorithms, and § 4 reports the protocol (LOOCV over 14 models, $C = 100$, bootstrap $B = 300$, fixed seed, 19 source benchmarks evaluated via lm-evaluation-harness and 4 agentic targets evaluated via the OpenHands SDK).

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release an anonymized code repository containing the selection, regression, and evaluation pipelines, along with the per-model source-benchmark and target-benchmark score matrices, accompanied by exact commands to reproduce Table 2 and Figure 1.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: § 4 specifies the LOOCV split, the budget C and bootstrap count B, the per-target ensemble weight tuning, and the regularization protocol; the methodology in § 3 defines the linear least-squares (Goal A) and logistic (Goal B) regressors and how their hyperparameters are auto-tuned via held-out evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: The headline LOOCV numbers in Table 2 are point estimates without error bars; we do, however, account for target-instance label noise via the bootstrap pooling described in § 3.1 and report a paired with-vs-without-bootstrap ablation in Table 3, which addresses the dominant source of variance in our regime.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The PACE regression and selection pipeline is light-weight (CPU-only; an SVD on a $14 \times 44{,}238$ matrix and per-target ridge fits) and runs in minutes on a commodity laptop; the upstream source-benchmark scoring is done with lm-evaluation-harness and the agentic targets with the OpenHands SDK, with per-model dollar costs reported in Figure 1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available benchmarks and frontier-model APIs in their intended evaluation regime; it does not involve human subjects, sensitive data, or release of high-risk artifacts.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: **??** discusses positive impacts—reducing the cost of agentic evaluation makes rigorous benchmarking more accessible and supports denser model-development monitoring—and notes that proxy predictions can drift under distribution shift, which could mislead deployment decisions if the proxy is over-trusted in place of full evaluation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released artifacts are evaluation utilities and aggregate score matrices over public benchmarks; we do not release any pretrained models, generative systems, or scraped datasets that pose misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All 19 source benchmarks and 4 agentic targets are cited at first mention in Table 1 and used under their respective public licenses; the lm-evaluation-harness and OpenHands SDK are likewise cited as the evaluation infrastructure.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The released code repository will include documentation for the PACE pipeline (selection, regression, evaluation), the per-model source/target score matrices, and example commands reproducing each table and figure in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects; all data are model evaluations on existing public benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the *subject* of evaluation in this work, not a component of the proposed method. The core PACE algorithm (SVD-based filter selection plus linear/logistic regression) does not involve LLMs in any non-standard way.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.