WHICH MODEL SHOULD BE CHOSEN? JOINT PERFORMANCE ESTIMATION AND RANKING OF BLACK-BOX MODELS ON UNLABELED DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Evaluating a pool of models on an unseen and unlabeled dataset requires both accurate performance estimates and reliable rankings. Many existing evaluation methods estimate performance for one model at a time, often requiring model-specific training or adaptation, access to training data, or internal representations to characterize distribution shifts. These requirements limit the evaluation of black-box models. Moreover, methods that evaluate models independently miss cross-model evidence that could improve ranking within a model pool. We propose PoolEvaluator, a framework for jointly estimating and ranking model performance without target labels or access to model internals. It estimates prior accuracies from relevant calibration datasets, then combines these priors with model agreement through expectation-maximization to infer latent correctness and refine each model's accuracy estimate using a closed-form update. An external budgeted judge is used later to refine ambiguous latent answers. Experiments across Text2SQL, image classification, and node classification show improved accuracy estimation and ranking with lower end-to-end evaluation latency than existing methods. This opens a new direction for jointly evaluating and ranking black-box model pools on unlabeled data through shared prediction evidence. Our code is available at `https://anonymous.4open.science/r/PoolEvaluator-664E`.

1 INTRODUCTION

Modern data environments are expanding rapidly, especially in large enterprises where systems continuously generate user activity, transactions, logs, and operational data. Meta (Karpathiotakis et al., 2025) reports ingesting more than 15 TB of new data per second, while Google BigQuery (Edara & Pasumansky, 2021) manages billions of data objects and is designed for workloads approaching hundreds of exabytes. This scale is reflected in recent benchmarks such as BIRD (Li et al., 2023) and Spider 2.0 (Lei et al., 2025) for Text2SQL, and DataComp (Gadre et al., 2023) for multimodal learning. However, reliable evaluation normally requires ground-truth labels, whose construction is costly (Yu et al., 2018; Boyeau et al., 2025), slow (Haas et al., 2015), and often constrained by enterprise privacy requirements (McMahan et al., 2017; Kairouz et al., 2021). As a result, new data may arrive before the previous evaluation set has been fully labeled.

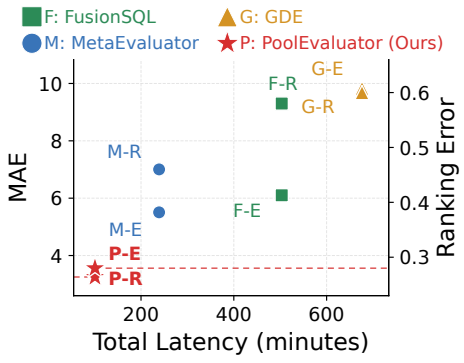


Figure 1: Avg. MAE (-E) and Avg. Kendall Ranking Error (-R) versus total evaluation time (minutes) for a 10-model pool in Text2SQL (training, adaptation, and inference). Lower is better for all axes.

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

At the same time, models are growing rapidly. Hugging Face (Yakefu et al., 2026) reports that public model repositories grew from 2.43 million to 2.96 million in the first seven months of 2026, or roughly 2,500 new repositories per day. This diversity spans multiple domains: (1) computer vision (Dosovitskiy et al., 2021; He et al., 2022; Radford et al., 2021; Liu et al., 2022), (2) graph learning (Kipf & Welling, 2017; Rampasek et al., 2022), and (3) Text2SQL (Pourreza & Rafiei, 2023; Gao et al., 2024; Li et al., 2024; 2025; Wang et al., 2025; Talaei et al., 2024; Pourreza et al., 2025; Lei et al., 2025). This raises a question: *How can we reliably estimate and rank the performance of each model in a model pool on a new unlabeled dataset?*

These challenges have motivated three approaches to label-free evaluation: (1) distribution-shift features (Deng & Zheng, 2021; Zheng et al., 2023; Pham et al., 2026b), (2) meta-learned evaluators (Pham et al., 2026a), and (3) disagreement among independently trained replicas through GDE (Jiang et al., 2022). The first two approaches require training data, model-specific evaluator training, or internal representations. GDE estimates error from how often independently trained copies of the same model produce different predictions. None of these methods directly ranks heterogeneous models within a pool. These requirements also limit their use for closed-source pools, and errors from separate evaluation runs can distort model rankings. As shown in Fig. 1, PoolEvaluator achieves lower MAE and Kendall ranking error than FusionSQL, MetaEvaluator, and GDE, while reducing total evaluation time: training, adaptation, and inference.

Given the limitations of existing methods, we propose PoolEvaluator, a framework that formulates label-free evaluation as a *joint inference* problem over *a pool of candidate models*. PoolEvaluator treats candidate models as heterogeneous annotators of the same unlabeled examples. It uses answers selected by consensus as initial *latent label* estimates, then applies EM to iteratively refine these estimates and model reliabilities, and calls an external judge to resolve ambiguous cases. PoolEvaluator reliably estimates model accuracies and rankings for model selection without access to the evaluated models' training data or internals. Our contributions are:

- **Formulation** To our knowledge, we are the first to formulate label-free model-pool evaluation where latent labels are estimated from the outputs of multiple candidate models.
- **Method** We present PoolEvaluator, a training-free EM framework that jointly refines latent label estimates and model reliabilities. We anchor reliability estimates with prior accuracy and iteratively update them until convergence. Then, an external judge is used to refine ambiguous latent answers.
- **Evaluation** We design the main evaluation around Text2SQL accuracy estimation and ranking, with complementary image and node classification experiments in the appendix to examine bounded label spaces.
- **Benchmarking** PoolEvaluator supports efficient benchmarking of model pools on unseen, unlabeled data. Across three modalities, it ranks models more reliably than existing methods, enabling robust top-k selection for deployment even when exact accuracy estimates remain uncertain.

2 Related Work

Current benchmarks and ranking systems mostly rely on task-specific ground truth for model evaluation, as in HumanEval (Chen et al., 2021a), MATH (Hendrycks et al., 2021b), and GSM8K (Cobbe et al., 2021). However, labeling new data is costly. Therefore, we study label-free evaluation across Text2SQL, image classification, and node classification, spanning relational, visual, and graph-structured data. This covers both the open-ended answer space of Text2SQL and the bounded label spaces of image and node classification, allowing us to test PoolEvaluator across both settings.

Text2SQL. Text2SQL evaluation relies on labeled benchmarks and execution-based metrics (Hong et al., 2025; Liu et al., 2025a), from Spider (Yu et al., 2018) to recent enterprise and live settings such as BIRD (Li et al., 2023), BEAVER (Chen et al., 2024a), LiveSQLBench (BIRD Team, 2025), and Spider 2.0 (Lei et al., 2025). FusionSQL (Pham et al., 2026b) trains an evaluator on shifts between source training data and synthetic datasets to predict the model's accuracy on unlabeled targets. After meta-learning across models and distribution shifts, MetaEvaluator (Pham et al., 2026a) transfers the learned knowledge by adapting a context vector for each unseen model.

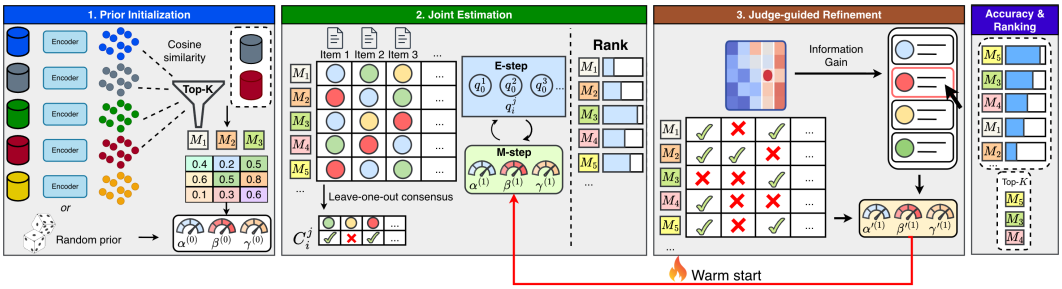


Figure 2: **Overview of PoolEvaluator.** *Stage 1:* PoolEvaluator selects the top-K labeled meta-dataset subsets most similar to the target and uses model accuracies on them to initialize α_j, β_j, and γ_j. If no such subsets exist or if there is no budget, initialization is random. *Stage 2:* Leave-one-out agreement yields C_i^j, and EM alternates between estimating q_i^j and updating the parameters. *Stage 3:* The highest-information item is judged by an LLM, after which EM resumes from the current estimates. Final accuracies rank the model pool.

Image classification. AutoEval (Deng & Zheng, 2021) trains an evaluator on transformed image datasets to learn how distribution shifts affect a classifier's accuracy on unlabeled data. MetaEvaluator (Pham et al., 2026a) also evaluates unseen image classifiers through model-specific adaptation. GDE (Jiang et al., 2022) relates test error to disagreement between independently trained classifiers. Other methods estimate accuracy from confidence shifts (Guillory et al., 2021; Garg et al., 2022), model agreement (Baek et al., 2022), or disagreement discrepancy (Rosenfeld & Garg, 2023).

Node classification. GOOD (Gui et al., 2022) reports substantial drops in GNN node-classification accuracy under distribution shifts. GNNEvaluator (Zheng et al., 2023) generates augmented source graphs and trains an evaluator on embedding-based shifts to predict a pretrained GNN's node-classification accuracy on unseen and unlabeled graphs.

These methods benefit when synthetic shifts cover target conditions, but require source training data, model-specific training or adaptation, and internal embeddings, limiting black-box deployment. Moreover, estimating each model's accuracy separately does not explicitly optimize cross-model relative rankings within a pool.

3 Problem Formulation

Therefore, we study the label-free evaluation of a model pool $\mathcal{M} = \{M_j\}_{j=1}^{J}$ *on an unlabeled target dataset* $\mathcal{X} = \{x_i\}_{i=1}^{N}$. For each target item x_i, model M_j produces an answer $r_i^j = M_j(x_i)$, while the unknown correct answer is denoted y_i. Each task provides an equivalence relation $\equiv_i$ that determines whether two answers to item i are equivalent. For classification, this corresponds to label equality, whereas for Text2SQL, equivalence may be defined in terms of execution equivalence. Let $\mathcal{A}_i$ denote the distinct answers produced by the models. Throughout this work, the indices i and j refer to a target item and a model, respectively.

GLAD. Majority voting (de Condorcet, 1785) gives every labeler equal weight. Dawid–Skene (Dawid & Skene, 1979) addresses this by learning a class-dependent confusion matrix for each labeler. However, the same matrix applies to every item of a class, so the labeler's response probability cannot vary with item difficulty. GLAD (Whitehill et al., 2009) instead makes correctness depend jointly on labeler expertise and item difficulty. Let α_j^{G} denote the expertise of model j and let $\beta_i^{\mathrm{G}} > 0$ denote the inverse difficulty of item i, whose difficulty is $1/\beta_i^{\mathrm{G}}$. In its binary formulation, the probability that model j is correct on item i is:

$$p_{ij}^{\mathrm{G}} := \mathbb{P}(r_i^j = y_i \mid \alpha_j^{\mathrm{G}}, \beta_i^{\mathrm{G}}) = \sigma(\alpha_j^{\mathrm{G}} \beta_i^{\mathrm{G}}),$$
$$\mathbb{P}(r_i^j \neq y_i \mid \alpha_j^{\mathrm{G}}, \beta_i^{\mathrm{G}}) = 1 - p_{ij}^{\mathrm{G}}, \qquad \sigma(u) = \frac{1}{1+\exp(-u)}.$$

GLAD jointly infers latent labels, model capability, and item difficulty with EM. Because $\alpha_j^{\mathrm{G}} \beta_i^{\mathrm{G}}$ appears inside the sigmoid function $\sigma(\cdot)$, its maximization has no closed-form solution and it is

optimized numerically. GLAD also does not explicitly parameterize false agreement, where multiple models return the same incorrect answer. Moreover, its canonical formulation does not directly accommodate open-ended, structured answers such as Text2SQL queries and their execution results.

PoolEvaluator. In our proposed framework, PoolEvaluator replaces fixed-class votes with answer equivalence and models agreement as evidence about correctness. We first define the latent correctness of model j on item i and its target accuracy: $Z_i^j = \mathbb{I}(r_i^j \equiv_i y_i)$ and $\alpha_j = \mathbb{P}(Z_i^j = 1)$. For model j, let $\mathcal{A}_i^{-j}$ denote the set of distinct answers produced by all other models, except model j itself. We define the leave-one-out consensus answer as the answer receiving the greatest support among these models and refer to it as the pseudo answer, $\widehat{y}_i^{-j} \in \operatorname{argmax}_{a \in \mathcal{A}_i^{-j}} \sum_{k \neq j} \mathbb{I}(r_i^k \equiv_i a)$. When answers have equal support, PoolEvaluator can break the tie by selecting the answer with the greatest accuracy-weighted support, $\sum_{k \neq j} \alpha_k^{(0)} \mathbb{I}(r_i^k \equiv_i a)$, where $\alpha_k^{(0)}$ is defined in Eq. 5. The leave-one-out construction makes $\widehat{y}_i^{-j}$ independent of model j's own output. We then define the agreement indicator between model j's response and the leave-one-out consensus as $C_i^j = \mathbb{I}(r_i^j \equiv_i \widehat{y}_i^{-j})$. We use β_j to characterize the probability of agreement when model j is correct and γ_j to characterize the probability of agreement when it is incorrect:

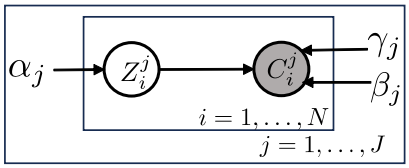


Figure 3: The proposed graphical model of PoolEvaluator.

$$Z_i^j \sim \text{Bernoulli}(\alpha_j), \quad C_i^j \mid Z_i^j = 1 \sim \text{Bernoulli}(\beta_j), \quad C_i^j \mid Z_i^j = 0 \sim \text{Bernoulli}(\gamma_j). \tag{1}$$

The graphical model of PoolEvaluator is summarized in Fig. 3. The corresponding joint distribution is given by:

$$p(\mathbf{C}, \mathbf{Z} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{j=1}^{J} p(Z_i^j \mid \alpha_j) p(C_i^j \mid Z_i^j, \beta_j, \gamma_j), \tag{2}$$

where $\mathbf{C} = \{C_i^j\}_{i,j}$ and $\mathbf{Z} = \{Z_i^j\}_{i,j}$. Let $\boldsymbol{\theta}_j = (\alpha_j, \beta_j, \gamma_j)^\mathsf{T}$ and let $\boldsymbol{\theta} = (\boldsymbol{\theta}_1^\mathsf{T}, \ldots, \boldsymbol{\theta}_J^\mathsf{T})^\mathsf{T}$ be the concatenated parameter vector.

In the original setting, $\mathbf{Z}$ is unobserved and must be inferred from the observed agreement variables ($\mathbf{C}$). To improve parameter estimation, we consider a semi-supervised setting in which the correctness of a subset of items is obtained using LLM judges. As demonstrated in the experimental section, incorporating this additional information can substantially improve estimation performance.

We obtain correctness labels from an LLM judge as follows. Let $\mathcal{L} \subseteq \{1, \ldots, N\}$ denote the judge-validated items and let $\mathcal{U} = \{1, \ldots, N\} \setminus \mathcal{L}$ denote the unjudged items. For each ambiguous item $i \in \mathcal{L}$, the judge selects $g_i \in \mathcal{A}_i$ or returns $g_i = \perp$ when no candidate is correct. This candidate-selection strategy avoids generating a new answer and follows the verifier-based candidate-selection approach (Cobbe et al., 2021; Chen et al., 2021b). Comparing the selected answer with each model's output then yields the judged correctness indicator:

$$z_i^j = \begin{cases} 1, & \text{if } g_i \neq \perp \text{ and } r_i^j \equiv_i g_i, \\ 0, & \text{otherwise,} \end{cases} \qquad i \in \mathcal{L}, \quad j = 1, \ldots, J. \tag{3}$$

Judged items use z_i^j, while unjudged items retain posterior correctness:

$$\widetilde{q}_i^j = \begin{cases} z_i^j, & i \in \mathcal{L}, \\ \mathbb{P}(Z_i^j = 1 \mid C_i^j, \boldsymbol{\theta}), & i \in \mathcal{U}. \end{cases}$$

4 The PoolEvaluator Framework

Fig. 2 illustrates the three stages of PoolEvaluator: prior initialization, joint estimation, and judge-guided refinement. To estimate the models' accuracies, we randomly initialize the parameters and use the EM algorithm for inference. However, because the likelihood function is non-convex, EM can be sensitive to parameter initialization. In practice, we can initialize the model-accuracy parameters using similar subsets from labeled meta-datasets (Pham et al., 2026a; Zheng et al., 2023).

EM then combines this prior information with the observed target-output agreements to effectively estimate the models' accuracies. We describe the initialization using public data in §4.1 and the EM updates for parameter estimation in §4.2.

Performance can be further improved in the semi-supervised setting, where an LLM judge is used to resolve selected ambiguities by choosing one of the candidate outputs or none. The details of this step are described in §4.3. Alg. 1 summarizes the overall three-stage procedure.

4.1 Stage 1: Prior Initialization

Let $\mathcal{X}$ be the unlabeled target dataset and $\mathcal{P} = \{\mathcal{P}_r\}_{r=1}^{R}$ be the collection of R labeled meta-dataset subsets (Pham et al., 2026a; Zheng et al., 2023). A fixed encoder ϕ represents each dataset by the mean of its item embeddings, and cosine similarity measures how closely each subset matches the target dataset:

$$\mathbf{h}(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}}\phi(x), \qquad s_r = \frac{\mathbf{h}(\mathcal{P}_r)^\top \mathbf{h}(\mathcal{X})}{\|\mathbf{h}(\mathcal{P}_r)\|_2 \|\mathbf{h}(\mathcal{X})\|_2}. \tag{4}$$

Under a budget of K subsets, we select the K highest similarities and pool their labeled items, $\mathcal{I}_K = \mathrm{TopK}_{r\in\{1,\dots,R\}}(s_r)$ and $\mathcal{D}_K = \bigcup_{r\in\mathcal{I}_K}\mathcal{P}_r$. Every calibration item $z \in \mathcal{D}_K$ carries a known answer $y(z)$. For Text2SQL, candidates are result tables from executable SQL queries.

Let $\mathrm{Acc}(M_j, \mathcal{P}_r)$ denote model j's measured accuracy on subset $\mathcal{P}_r$. With smoothing $\lambda > 0$, we initialize the accuracy by the subset-averaged accuracy, and the two agreement rates by their empirical frequencies among correct and incorrect responses:

$$\alpha_j^{(0)} = \frac{1}{K}\sum_{r\in\mathcal{I}_K}\mathrm{Acc}(M_j,\mathcal{P}_r), \quad \beta_j^{(0)} = \frac{\lambda + \sum_{z\in\mathcal{D}_K} Z_z^j C_z^j}{2\lambda + \sum_{z\in\mathcal{D}_K} Z_z^j}, \quad \gamma_j^{(0)} = \frac{\lambda + \sum_{z\in\mathcal{D}_K}(1-Z_z^j) C_z^j}{2\lambda + \sum_{z\in\mathcal{D}_K}(1-Z_z^j)}. \tag{5}$$

Here $\beta_j^{(0)}$ and $\gamma_j^{(0)}$ are the agreement rates when model j is correct and incorrect, respectively, and $\boldsymbol{\theta}_j^{(0)} = (\alpha_j^{(0)}, \beta_j^{(0)}, \gamma_j^{(0)})^\top$. We set $\lambda = 1$ to keep both rates in $(0,1)$ even when all or no calibration answers are correct. Informative initialization can guide EM toward a better local optimum (Biernacki et al., 2003; Karlis & Xekalaki, 2003). When $K = 0$, each coordinate of $\boldsymbol{\theta}^{(0)}$ is instead drawn independently from a normal distribution truncated to $(0,1)$. Fig. 7 compares both initializations.

4.2 Stage 2: Joint Estimation

At iteration t, the E-step computes $q_i^j = \mathbb{P}(Z_i^j = 1 \mid C_i^j, \boldsymbol{\theta}_j^{(t)})$, the probability that model j is correct on item i:

$$q_i^j = \frac{\alpha_j^{(t)}(\beta_j^{(t)})^{C_i^j}(1-\beta_j^{(t)})^{1-C_i^j}}{\alpha_j^{(t)}(\beta_j^{(t)})^{C_i^j}(1-\beta_j^{(t)})^{1-C_i^j} + (1-\alpha_j^{(t)})(\gamma_j^{(t)})^{C_i^j}(1-\gamma_j^{(t)})^{1-C_i^j}}. \tag{6}$$

The M-step updates model accuracy, agreement when correct, and false agreement when incorrect:

$$\alpha_j^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} q_i^j, \qquad \beta_j^{(t+1)} = \frac{\sum_{i=1}^{N} q_i^j C_i^j}{\sum_{i=1}^{N} q_i^j}, \qquad \gamma_j^{(t+1)} = \frac{\sum_{i=1}^{N}(1-q_i^j) C_i^j}{\sum_{i=1}^{N}(1-q_i^j)}. \tag{7}$$

The final α_j estimates provide the model accuracies and their ranking. §A.1 gives the complete derivation. Unlike our closed-form updates, GLAD (Whitehill et al., 2009) uses gradient ascent in every M-step, adding an iterative optimization loop that increases computation and finds only a local maximizer. This is demonstrated in Appendix D.

4.3 Stage 3: Judge-Guided Refinement

After Stage 2, the judge validates a budgeted set $\mathcal{L}$, leaving $\mathcal{U} = \{1,\dots,N\} \setminus \mathcal{L}$ unjudged. For each candidate $i \in \mathcal{U}$, let $\mathcal{Y}_i = \mathcal{A}_i \cup \{\perp\}$ contain the distinct model answers and $\perp$ when none is correct. For an answer $a \neq \perp$, let $\mathcal{J}_i(a) = \{j : r_i^j \equiv_i a\}$ contain its supporting models. Under the factorized-posterior approximation, answer a requires every supporting model to be correct and

Table 1: MAE in percentage points (mean ± 95% CI over 10 runs, lower is better). Pool size is 10 models. Each column is a target unlabeled dataset. Avg. is the mean across the seven datasets, with the CI averaged over columns. Best results are **bold** and second-best results are underlined.

Method	Spider	BIRD	Spider 2.0	BEAVER	ScienceBench	EntSQL	LiveSQLBench	Avg.
ConfProfile	8.46±0.94	9.71±1.55	11.95±1.71	11.20±1.64	10.38±1.58	11.10±1.57	11.42±1.70	10.60±1.53
ALine-D	8.32±1.09	9.80±1.24	11.61±1.74	10.95±1.82	10.07±1.39	10.37±1.70	11.40±1.92	10.36±1.56
NL2SQL-BUGs	8.22±1.27	9.31±1.39	11.35±1.67	11.17±1.48	9.98±1.61	10.72±1.33	11.43±1.54	10.31±1.47
ATC	7.93±0.99	9.53±1.40	11.88±1.71	10.88±1.56	9.63±1.58	10.56±1.38	11.37±1.52	10.25±1.45
DoC	7.71±1.06	9.44±1.23	11.41±1.74	10.49±1.74	9.75±1.63	9.85±1.48	10.82±1.55	9.92±1.49
GDE	7.59±0.98	8.89±1.39	11.12±1.98	10.60±1.77	9.65±1.62	9.91±1.44	11.06±1.87	9.83±1.58
FusionSQL	5.25±1.26	5.99±1.21	6.69±1.50	6.45±1.54	5.97±1.17	6.27±1.31	6.09±1.56	6.10±1.36
MetaEvaluator	4.55±0.87	5.09±1.34	6.39±1.91	6.15±1.58	5.57±1.33	4.26±1.71	6.53±1.90	5.51±1.52
PoolEvaluator	**2.23±0.49**	**3.10±0.86**	**3.93±1.41**	**3.50±1.24**	**2.85±0.85**	**3.39±1.01**	**3.76±1.22**	**3.25±1.01**

every model $k \notin \mathcal{J}_i(a)$ to be wrong. Therefore, the posterior q_i^j from Eq. 6 gives the answer support $\rho_i(a)$. Normalizing over $\mathcal{Y}_i$ conditions on these feasible correctness vectors and gives $\pi_i(a)$:

$$\rho_i(a) = \prod_{j \in \mathcal{J}_i(a)} q_i^j \prod_{k \notin \mathcal{J}_i(a)} (1 - q_i^k), \quad a \neq \perp,$$

$$\rho_i(\perp) = \prod_{j=1}^{J} (1 - q_i^j), \qquad \pi_i(a) = \frac{\rho_i(a)}{\rho_i(\perp) + \sum_{a' \in \mathcal{A}_i} \rho_i(a')}, a \in \mathcal{Y}_i.$$

Let $g_i \in \mathcal{Y}_i$ denote the judge's answer. Let $\mathcal{H}(\mathcal{U})$ be the current posterior entropy and let $\mathcal{H}_i^a$ be the remaining entropy after hypothetically observing $g_i = a$ and updating EM. We score the unjudged items and select:

$$\mathrm{IG}(i) = \mathcal{H}(\mathcal{U}) - \sum_{a \in \mathcal{Y}_i} \pi_i(a) \mathcal{H}_i^a, \qquad i^\star = \operatorname*{argmax}_{i \in \mathcal{U}} \mathrm{IG}(i). \tag{8}$$

Thus, each round validates the single item expected to remove the most uncertainty, provided that $\mathrm{IG}(i^\star) > 0$. The derivation in §A.3 gives Eq. 8. Using the observed correctness z_i^j from Eq. 3, the next E-step replaces each judged posterior with its observed value:

$$\widetilde{q}_i^j = \begin{cases} z_i^j, & i \in \mathcal{L}, \\ q_i^j, & i \in \mathcal{U}, \end{cases} \tag{9}$$

where q_i^j is given by Eq. 6. Substituting these values into Eq. 7 preserves the closed-form M-step:

$$\alpha_j^{(t+1)} = \frac{\sum_{i \in \mathcal{L}} z_i^j + \sum_{i \in \mathcal{U}} q_i^j}{N}, \qquad \beta_j^{(t+1)} = \frac{\sum_{i \in \mathcal{L}} z_i^j C_i^j + \sum_{i \in \mathcal{U}} q_i^j C_i^j}{\sum_{i \in \mathcal{L}} z_i^j + \sum_{i \in \mathcal{U}} q_i^j},$$
$$\gamma_j^{(t+1)} = \frac{\sum_{i \in \mathcal{L}} (1 - z_i^j) C_i^j + \sum_{i \in \mathcal{U}} (1 - q_i^j) C_i^j}{\sum_{i \in \mathcal{L}} (1 - z_i^j) + \sum_{i \in \mathcal{U}} (1 - q_i^j)}. \tag{10}$$

§A.2 derives these updates. Correct validation cannot increase uncertainty, with a strict decrease in some parameter direction when a validated pair previously had $0 < q_i^j < 1$, as proven in §A.3. At judge round v, we warm-start EM from the previously converged parameters $\widehat{\boldsymbol{\theta}}_v$. Validating one of the $N = |\mathcal{X}|$ target items moves the updated local optimum by at most $O(1/N)$, as derived in §A.4. This stable shift accelerates convergence in our experiments. The judge selects a candidate answer or returns none rather than generating a new answer, since candidate verification can improve reasoning reliability (Weng et al., 2023; Kadavath et al., 2022; Kawabata & Sugawara, 2024; Saad-Falcon et al., 2025).

5 Experiments

We focus on Text2SQL because its open-ended output space has no fixed class vocabulary, making unlabeled evaluation particularly challenging. PoolEvaluator supports heterogeneous black-box pools using only model outputs. For comparability with existing baselines, we use open-source models and datasets. The evaluated model pool is listed in Appendix C. The results for image and node classification are reported separately in Appendix D and Appendix E.

Algorithm 1 PoolEvaluator

Require: Target data $\mathcal{X}$, meta-dataset subsets $\mathcal{P}$, models $\mathcal{M}$, subset budget $K \geq 0$, judge-round budget $V \geq 0$, EM limit T, tolerance ε

▷ **Stage 1: Prior initialization**

if $K > 0$ **then**
  Retrieve the top-K subsets as described in Eq. 4
  Evaluate each selected subset and initialize $\boldsymbol{\theta}^{(0)}$ as described in Eq. 5
else
  Draw each coordinate of $\boldsymbol{\theta}^{(0)}$ independently from a normal distribution truncated to $(0, 1)$
end if

▷ **Stage 2: Joint reliability estimation**

Run the models on $\mathcal{X}$ and compute the leave-one-out agreements $\{C_i^j\}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
for $t = 0, \ldots, T-1$ **do**
  $\boldsymbol{\theta}_{\text{old}} \leftarrow \boldsymbol{\theta}$
  Compute $\{q_i^j\}$ by the E-step using Eq. 6
  Update $\boldsymbol{\theta}$ by the M-step using Eq. 7
  if $\|\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{old}}\|_\infty < \varepsilon$ **then**
    break
  end if
end for

▷ **Stage 3: Optional judge-guided refinement**

$\mathcal{L} \leftarrow \varnothing$ and $\mathcal{U} \leftarrow \{1, \ldots, N\}$
for $v = 1, \ldots, V$ **do**
  if $\mathcal{U} = \varnothing$ **then**
    break
  end if
  Select $i^\star$ using Eq. 8
  if $\mathrm{IG}(i^\star) \leq 0$ **then**
    break
  end if
  Query the judge on item $i^\star$ and compute its judged correctness indicators using Eq. 3
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{i^\star\}$ and $\mathcal{U} \leftarrow \mathcal{U} \setminus \{i^\star\}$
  Warm-start EM from the current $\boldsymbol{\theta}$
  for $t = 0, \ldots, T-1$ **do**
    $\boldsymbol{\theta}_{\text{old}} \leftarrow \boldsymbol{\theta}$
    Compute $\{\tilde{q}_i^j\}$ using Eq. 9
    Update $\boldsymbol{\theta}$ using Eq. 10
    if $\|\boldsymbol{\theta} - \boldsymbol{\theta}_{\text{old}}\|_\infty < \varepsilon$ **then**
      break
    end if
  end for
end for
return final accuracy estimates $\{\alpha_j\}$ and their descending ranking

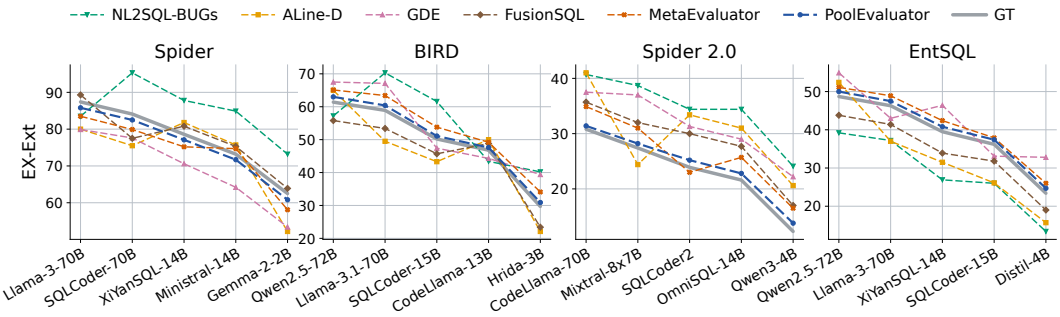


Figure 4: Calibration of accuracy estimation across four Text2SQL targets. Each panel plots average estimated EX-Ext against the ground truth (GT) for a pool of 10 models, showing five representative models per panel. Curves closer to GT indicate better-calibrated estimates.

5.1 Experimental Setup

Metrics and Implementation Details. We use EX-Extended (EX-Ext, Appendix B), which requires matching results on the original database and five modified instances to reduce spurious EX matches. We report MAE against true EX-Ext, pairwise ordering accuracy, top-k selection accuracy, Kendall τ, and end-to-end evaluation latency. Text2SQL uses $K = 15$ subsets and $V = 10$ judge rounds, each validating one item. Unless stated otherwise, we average the outputs of GPT-5.4, GPT-5.5, and

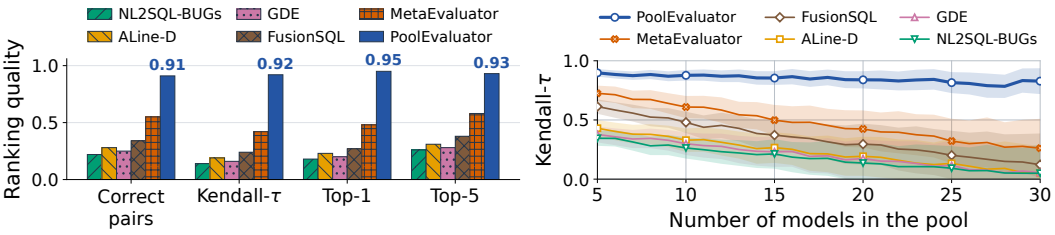


Figure 5: *Left*: ranking quality averaged over pools of 5 to 30 models. Higher is better for all four metrics. *Right*: Kendall's τ against pool size, with shaded 95% CIs over repeated runs.

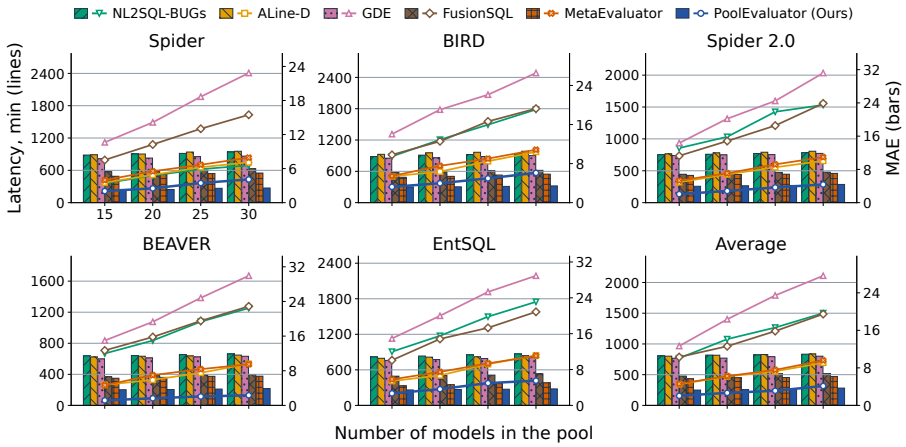


Figure 6: MAE and total evaluation latency across pool sizes. The average is computed over all seven datasets in Tab. 1. Lines use the left axis and bars use the right axis. Lower is better.

Claude Opus 4.5 as the LLM judges. Experiments run on four 24 GB NVIDIA RTX 4090 GPUs and an Intel Core i7-14700 CPU. Evaluator training and adaptation use bfloat16 mixed precision and parallel data loading with fixed random seeds to support reproducibility.

Datasets and Baselines. We evaluate on Spider (Yu et al., 2018), BIRD Bench (Li et al., 2023), Spider 2.0 (Lei et al., 2025), BEAVER (Chen et al., 2024a), ScienceBenchmark (Zhang et al., 2023), EntSQL (Liao et al., 2026), and LiveSQLBench (BIRD Team, 2025). Target labels are hidden from every evaluator and used only afterward for evaluation. We compare PoolEvaluator with FusionSQL (Pham et al., 2026b), MetaEvaluator (Pham et al., 2026a), NL2SQL-BUGs (Liu et al., 2025b), GDE (Jiang et al., 2022), ALine-D (Baek et al., 2022), ATC (Garg et al., 2022), DoC (Guillory et al., 2021), and ConfProfile (Fu et al., 2023).

5.2 Performance Comparison

Evaluation Accuracy. As shown in Tab. 1, PoolEvaluator achieves the lowest MAE across all datasets and the narrowest average 95% CI. Errors increase on harder benchmarks such as Spider 2.0, BEAVER, and LiveSQLBench, where each model's performance is worse and varies more across sampled model pools. However, Fig. 4 shows that PoolEvaluator tracks ground-truth performance consistently across models, scales, and datasets. PoolEvaluator slightly underestimates model performance on Spider and overestimates it on harder datasets. In contrast, other methods show larger and less systematic deviations across models, which can distort pairwise differences and lead to ranking errors. This is shown in later experimental results. Tab. 1 and Fig. 4 average over 10 repeated runs, each sampling a pool of 10 models from Appendix C.

Ranking Stability. Fig. 5 shows that PoolEvaluator consistently achieves the best ranking quality across correct pairs, Kendall's τ, Top-1, and Top-5. As the pool grows from 5 to 30 models, its Kendall's τ remains high with narrow 95% CIs, while all baselines degrade more substantially. This indicates that PoolEvaluator preserves model ordering more reliably as pool size increases. This is increasingly important in practice, where rapidly expanding datasets and model pools can outpace

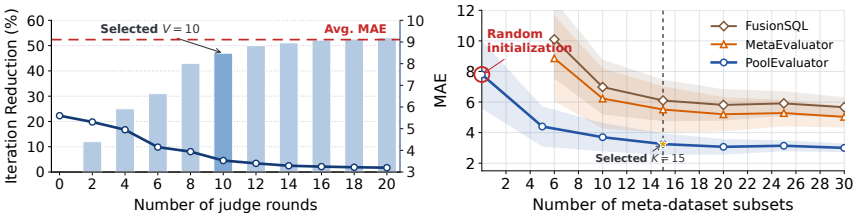


Figure 7: *Left*: iteration reduction (bars, left axis) and MAE (blue line, right axis) across single-item judge rounds. *Right*: MAE by the number of selected meta-dataset subsets. Shading denotes 95% CIs. The red line is the average MAE of baseline methods in Tab. 1.

manual labeling, making reliable model ranking essential for selecting top-performing models for deployment even when absolute performance estimates might be imperfect.

5.3 Evaluation Efficiency

Pool Scalability. Fig. 6 shows that PoolEvaluator maintains the lowest MAE and total evaluation latency across different datasets and pool sizes. Baselines require evaluator training or adaptation, and their longer runtime and higher MAE reduce efficiency and ranking quality, consistent with the right panel of Fig. 5. NL2SQL-BUGs uses an LLM judge for every sample from every model. This cost is moderate on Spider but rises sharply on harder targets such as Spider 2.0, BEAVER, and EntSQL, which require longer or repeated verification. In contrast, PoolEvaluator uses a fixed budget to validate only ambiguous items selected by the expected information gain in Eq. 8, as analyzed next.

Judge Budget. Following the selective strategy in Eq. 8, the left panel of Fig. 7 shows the percentage of EM iterations saved by warm-starting from the previous solution and the resulting MAE. At $V = 0$, no iteration reduction is available, but PoolEvaluator's MAE is already below the competing-method average. Iteration reduction reaches approximately 47% at $V = 10$, while iteration reduction and MAE improve more slowly after 12–14 rounds. Therefore, we use $V = 10$ to balance estimation quality and validation cost. These results also support delayed annotation: when human review lags behind model development, or privacy constraints postpone label access, newly available labels can be incorporated later to improve performance estimates and reliably re-rank the model pool.

Prior Budget. The right panel of Fig. 7 examines how the number of meta-dataset subsets chosen affects MAE, with $K = 0$ indicating random prior initialization for PoolEvaluator. FusionSQL and MetaEvaluator use subset accuracies to train evaluators across distribution shifts, whereas PoolEvaluator uses them only to initialize EM. Even with $K = 0$, PoolEvaluator outperforms both baselines. All three methods incur the same subset evaluation cost, but PoolEvaluator achieves lower MAE with far fewer subsets and plateaus by $K = 15$. Therefore, we use $K = 15$ throughout the experiments. More generally, the number of subsets can be adjusted to meet a target MAE, enabling budgeted evaluation under different cost-accuracy trade-offs.

6 Discussion & Conclusion

We studied model-pool evaluation on unseen, unlabeled data, where datasets and models can grow faster than human annotation. PoolEvaluator retrieves target-relevant subsets to initialize model reliabilities, combines these priors with shared-output evidence through closed-form EM, and selectively validates ambiguous items under a judge budget. Using only black-box model outputs, it improves estimation accuracy and ranking stability while reducing evaluation latency across Text2SQL, image classification, and node classification. Even at zero budget ($K = 0$ and $V = 0$), PoolEvaluator remains competitive, outperforms most baselines, and requires no evaluator training. The judge-guided refinement stage shows that labels obtained later can refine the estimates and re-rank the pool, supporting delayed annotation when human review or privacy constraints postpone label access. Future work can study batched judge validation and extend this framework to open-ended tasks such as image captioning and subgraph matching.

AI USE STATEMENT

We used generative AI only to polish text drafted by the authors. It was not used for research ideation, theoretical derivation or verification, or literature retrieval and summarization. The authors reviewed and edited all AI-assisted text and take full responsibility for the manuscript, its claims, and its artifacts.

ETHICS STATEMENT

We evaluate Text2SQL, image classification, and node classification models on public benchmarks. This study involves no human subjects, personal information, or sensitive data. PoolEvaluator estimates accuracy from agreement within a model pool. Its estimates should support, not replace, labeled validation.

REPRODUCIBILITY STATEMENT

We state the assumptions, propositions, estimator, and inference procedure in §3 and §4. §5 describes the model pools, benchmarks, calibration data, and evaluation protocol. Our code and scripts for reproducing the experiments are available at `https://anonymous.4open.science/r/PoolEvaluator-664E`.

REFERENCES

Christina Baek, Yiding Jiang, Aditi Raghunathan, and J. Zico Kolter. Agreement-on-the-line: Predicting the performance of neural networks under distribution shift. In *NeurIPS*, 2022.

Christophe Biernacki, Gilles Celeux, and Gérard Govaert. Choosing starting values for the EM algorithm for getting the highest likelihood in multivariate Gaussian mixture models. *Computational Statistics & Data Analysis*, 2003.

BIRD Team. LiveSQLBench: A dynamic and contamination-free benchmark for evaluating LLMs on real-world text-to-SQL tasks. `https://livesqlbench.ai/`, 2025.

Pierre Boyeau, Anastasios Nikolas Angelopoulos, Tianle Li, Nir Yosef, Jitendra Malik, and Michael I. Jordan. AutoEval done right: Using synthetic data for model evaluation. In *ICML*, 2025.

Mark Chen et al. Evaluating large language models trained on code. *arXiv:2107.03374*, 2021a.

Peter Baile Chen, Devin Yang, Weiyue Li, Fabian Wenz, Yi Zhang, Nesime Tatbul, Michael Cafarella, Çağatay Demiralp, and Michael Stonebraker. BEAVER: An enterprise benchmark for text-to-SQL. *arXiv:2409.02038*, 2024a.

Runjin Chen, Tong Zhao, Ajay Kumar Jaiswal, Neil Shah, and Zhangyang Wang. LLaGA: Large language and graph assistant. In *ICML*, 2024b.

Sihao Chen, Fan Zhang, Kazoo Sone, and Dan Roth. Improving faithfulness in abstractive summarization with contrast candidate generation and selection. In *NAACL*, 2021b.

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

Alexander Philip Dawid and Allan M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *JRSS C*, 1979.

Marquis de Condorcet. *Essai sur l'application de l'analyse a la probabilite des decisions rendues a la pluralite des voix*. Imprimerie Royale, 1785.

Weijian Deng and Liang Zheng. Are labels always necessary for classifier accuracy evaluation? In *CVPR*, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Pavan Edara and Mosha Pasumansky. Big metadata: When metadata is big data. *PVLDB*, 2021.

Harvey Fu, Qinyuan Ye, Albert Xu, Xiang Ren, and Robin Jia. Estimating large language model capabilities without labeled test data. In *Findings of EMNLP*, 2023.

Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, et al. DataComp: In search of the next generation of multimodal datasets. In *NeurIPS*, 2023.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-SQL empowered by large language models: A benchmark evaluation. *PVLDB*, 2024.

Saurabh Garg, Sivaraman Balakrishnan, Zachary C. Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *ICLR*, 2022.

Shurui Gui, Xiner Li, Limei Wang, and Shuiwang Ji. GOOD: A graph out-of-distribution benchmark. In *NeurIPS*, 2022.

Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *ICCV*, 2021.

Daniel Haas, Jiannan Wang, Eugene Wu, and Michael J. Franklin. CLAMShell: Speeding up crowds for low-latency data labeling. *PVLDB*, 2015.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 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. In *NeurIPS*, 2021b.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-generation database interfaces: A survey of LLM-based text-to-SQL. *TKDE*, 2025.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

Jonathan J. Hull. A database for handwritten text recognition research. *TPAMI*, 1994.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J. Zico Kolter. Assessing generalization of SGD via disagreement. In *ICLR*, 2022.

Saurav Kadavath et al. Language models mostly know what they know. *arXiv:2207.05221*, 2022.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021.

Dimitris Karlis and Evdokia Xekalaki. Choosing initial values for the EM algorithm for finite mixtures. *Computational Statistics & Data Analysis*, 2003.

Manos Karpathiotakis, Vlassios Rizopoulos, Basri Kahveci, Tiziano Carotti, Artem Gelum, Hazem Nada, and Yuri Dolgov. Scribe: How meta transports terabytes per second in real time. *PVLDB*, 2025.

Akira Kawabata and Saku Sugawara. Rationale-aware answer verification by pairwise self-evaluation. In *EMNLP*, 2024.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *ICLR*, 2025.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. CodeS: Towards building open-source language models for text-to-SQL. *PACMMOD*, 2024.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. OmniSQL: Synthesizing high-quality text-to-SQL data at scale. *PVLDB*, 2025.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.-C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. In *NeurIPS*, 2023.

Chengxi Liao, Tao Xu, Zulong Chen, Chuanfei Xu, Yiyan Wang, Xinyun Wang, Yanlong Zhang, Xiaojun Chen, Zhibo Yang, and Zeyi Wen. EntSQL: A benchmark for grounding text-to-SQL in long-context enterprise knowledge. *arXiv:2606.03363*, 2026.

Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuxin Zhang, Ju Fan, Guoliang Li, Nan Tang, and Yuyu Luo. A survey of text-to-SQL in the era of LLMs: Where are we, and where are we going? *TKDE*, 2025a.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. NL2SQL-BUGs: A benchmark for detecting semantic errors in NL2SQL translation. In *KDD*, 2025b.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, 2022.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Trinh Pham, Viet Huynh, Hongzhi Yin, Quoc Viet Hung Nguyen, and Thanh Tam Nguyen. Learning to evaluate: Cost-effective model evaluation on unlabeled data with meta-learning. In *KDD*, 2026a.

Trinh Pham, Thanh Tam Nguyen, Viet Huynh, Hongzhi Yin, and Quoc Viet Hung Nguyen. An efficient and effective evaluator for Text2SQL models on unseen and unlabeled data. In *ICDE*, 2026b.

Mohammadreza Pourreza and Davood Rafiei. DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction. In *NeurIPS*, 2023.

Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. CHASE-SQL: Multi-path reasoning and preference optimized candidate selection in text-to-SQL. In *ICLR*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Ladislav Rampasek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*, 2022.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *JMLR*, 2010.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to CIFAR-10? *arXiv:1806.00451*, 2018.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet classifiers generalize to ImageNet? In *ICML*, 2019.

Elan Rosenfeld and Saurabh Garg. (almost) provable error bounds under distribution shift via disagreement discrepancy. In *NeurIPS*, 2023.

Jon Saad-Falcon, Estefany Kelly Buchanan, Mayee Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Ré. Weaver: Shrinking the generation–verification gap by scaling compute for verification. In *NeurIPS*, 2025.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. CHESS: Contextual harnessing for efficient SQL synthesis. *arXiv:2405.16755*, 2024.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. GraphGPT: Graph instruction tuning for large language models. In *SIGIR*, 2024.

Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In *COLING*, 2025.

Haohan Wang, Songwei Ge, Zachary C. Lipton, and Eric P. Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, 2019.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *EMNLP Findings*, 2023.

Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. In *NeurIPS*, 2009.

Adina Yakefu, Apolinário, and Irene Solaiman. State of open models: Summer 2026 observations. Hugging Face, 2026.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *EMNLP*, 2018.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023.

Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. ScienceBenchmark: A complex real-world benchmark for evaluating natural language to SQL systems. *PVLDB*, 2023.

Yuchen Zhang, Xi Chen, Dengyong Zhou, and Michael I. Jordan. Spectral methods meet EM: A provably optimal algorithm for crowdsourcing. *JMLR*, 2016.

Xin Zheng, Miao Zhang, Chunyang Chen, Soheila Molaei, Chuan Zhou, and Shirui Pan. GNNEvaluator: Evaluating GNN performance on unseen graphs without labels. In *NeurIPS*, 2023.

APPENDIX CONTENTS

A Formulation Derivations **15**

A.1 Closed-Form EM Derivation 15

A.2 Judge-Conditioned EM Updates 16

A.3 Uncertainty Reduction After Judge Validation 17

A.4 Warm Starts After Judge Validation 18

B EX and EX-Ext **19**

C Text2SQL Model Pool **20**

D Image Classification **20**

E Node Classification **22**

A Formulation Derivations

A.1 Closed-Form EM Derivation

Condition on the observed agreement matrix $\mathbf{C}$. Under the working pairwise factorization in Eq. 2, substituting the Bernoulli probabilities from Eq. 1 into Bayes' rule gives the E-step in Eq. 6. With $q_i^j = \mathbb{E}[Z_i^j \mid C_i^j, \boldsymbol{\theta}^{(t)}]$ fixed, the expected complete-data log-likelihood is:

$$\begin{aligned} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}^{(t)}) = \sum_{i=1}^{N}\sum_{j=1}^{J} \big[& q_i^j \log \alpha_j + (1-q_i^j)\log(1-\alpha_j) \\ & + q_i^j \{C_i^j \log \beta_j + (1-C_i^j)\log(1-\beta_j)\} \\ & + (1-q_i^j)\{C_i^j \log \gamma_j + (1-C_i^j)\log(1-\gamma_j)\}\big]. \end{aligned} \tag{11}$$

The three parameters of model j appear in separate additive groups of Eq. 11, and no term couples two of them. Therefore, each can be maximized on its own.

Update for α_j. Collecting the terms that contain α_j and differentiating gives:

$$Q_{\alpha_j} = \sum_{i=1}^{N} \left[q_i^j \log \alpha_j + (1-q_i^j)\log(1-\alpha_j) \right], \qquad \frac{\partial Q_{\alpha_j}}{\partial \alpha_j} = \sum_{i=1}^{N} \left[\frac{q_i^j}{\alpha_j} - \frac{1-q_i^j}{1-\alpha_j} \right].$$

Setting the derivative to zero and multiplying by $\alpha_j(1-\alpha_j) > 0$ gives $(1-\alpha_j)\sum_i q_i^j = \alpha_j \sum_i (1-q_i^j)$. Since $\sum_i (1-q_i^j) = N - \sum_i q_i^j$, the terms in $\alpha_j \sum_i q_i^j$ cancel and:

$$\sum_{i=1}^{N} q_i^j = \alpha_j N \qquad \Longleftrightarrow \qquad \alpha_j^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} q_i^j. \tag{12}$$

Update for β_j. The terms that contain β_j carry the weight q_i^j, because agreement is modeled by β_j only in the correct state:

$$Q_{\beta_j} = \sum_{i=1}^{N} q_i^j \left[C_i^j \log \beta_j + (1-C_i^j)\log(1-\beta_j) \right], \qquad \frac{\partial Q_{\beta_j}}{\partial \beta_j} = \sum_{i=1}^{N} q_i^j \left[\frac{C_i^j}{\beta_j} - \frac{1-C_i^j}{1-\beta_j} \right].$$

Setting the derivative to zero and multiplying by $\beta_j(1-\beta_j) > 0$ gives $(1-\beta_j)\sum_i q_i^j C_i^j = \beta_j \sum_i q_i^j (1-C_i^j)$. Expanding the right-hand side as $\beta_j \sum_i q_i^j - \beta_j \sum_i q_i^j C_i^j$ cancels the terms in $\beta_j \sum_i q_i^j C_i^j$ and leaves:

$$\sum_{i=1}^{N} q_i^j C_i^j = \beta_j \sum_{i=1}^{N} q_i^j \qquad \Longleftrightarrow \qquad \beta_j^{(t+1)} = \frac{\sum_{i=1}^{N} q_i^j C_i^j}{\sum_{i=1}^{N} q_i^j}. \tag{13}$$

Update for γ_j. The same argument applies in the incorrect state, where the weight is $1 - q_i^j$:

$$\begin{aligned} Q_{\gamma_j} &= \sum_{i=1}^{N} (1-q_i^j) \left[C_i^j \log \gamma_j + (1-C_i^j)\log(1-\gamma_j) \right], \\ \frac{\partial Q_{\gamma_j}}{\partial \gamma_j} &= \sum_{i=1}^{N} (1-q_i^j) \left[\frac{C_i^j}{\gamma_j} - \frac{1-C_i^j}{1-\gamma_j} \right]. \end{aligned}$$

Setting the derivative to zero and canceling as before gives:

$$\gamma_j^{(t+1)} = \frac{\sum_{i=1}^{N} (1-q_i^j) C_i^j}{\sum_{i=1}^{N} (1-q_i^j)}. \tag{14}$$

Together, Eq. 12, Eq. 13, and Eq. 14 are the three updates of Eq. 7. Each is a weighted agreement frequency: α_j is the average posterior correctness, β_j is the agreement rate weighted by the posterior of being correct, and γ_j is the agreement rate weighted by the posterior of being wrong.

Maximality. Every objective has the form $G(p) = A \log p + B \log(1-p)$ with $A, B \geq 0$. Its second derivative is:

$$G''(p) = -\frac{A}{p^2} - \frac{B}{(1-p)^2} \leq 0.$$

Therefore, each objective is concave on $(0, 1)$, so the stationary point found is its global maximizer.

A.2 Judge-Conditioned EM Updates

We first prove the judge-conditioned E-step and M-step. Let $\mathcal{L}$ contain the judged items and let $\mathcal{U} = \{1, \ldots, N\} \setminus \mathcal{L}$ contain the remaining items. The judged pairs (C_i^j, z_i^j) are complete observations, while Z_i^j remains latent on $\mathcal{U}$. The corresponding observed-data log-likelihood is:

$$\ell_{\text{ss}}(\boldsymbol{\theta}) = \sum_{j=1}^{J} \sum_{i \in \mathcal{L}} \log p(C_i^j, z_i^j \mid \boldsymbol{\theta}_j) + \sum_{j=1}^{J} \sum_{i \in \mathcal{U}} \log \sum_{z \in \{0,1\}} p(C_i^j, z \mid \boldsymbol{\theta}_j).$$

For $i \in \mathcal{L}$, exact validation makes the conditional distribution of Z_i^j a point mass at z_i^j. For $i \in \mathcal{U}$, the original posterior remains q_i^j. Therefore, the conditional expectation is exactly $\widetilde{q}_i^j = z_i^j$ on $\mathcal{L}$ and $\widetilde{q}_i^j = q_i^j$ on $\mathcal{U}$, which proves Eq. 9. For a fixed model j, taking the conditional expectation of the complete-data log-likelihood gives:

$$\begin{aligned} Q_j = &\sum_{i=1}^{N} \left[\widetilde{q}_i^j \log \alpha_j + (1 - \widetilde{q}_i^j) \log(1 - \alpha_j) \right] \\ &+ \sum_{i=1}^{N} \widetilde{q}_i^j \left[C_i^j \log \beta_j + (1 - C_i^j) \log(1 - \beta_j) \right] \\ &+ \sum_{i=1}^{N} (1 - \widetilde{q}_i^j) \left[C_i^j \log \gamma_j + (1 - C_i^j) \log(1 - \gamma_j) \right]. \end{aligned}$$

Define the expected total and agreement counts as follows:

$$\begin{aligned} A_\alpha &= \sum_{i \in \mathcal{L}} z_i^j + \sum_{i \in \mathcal{U}} q_i^j, \\ T_\beta &= A_\alpha, & A_\beta &= \sum_{i \in \mathcal{L}} z_i^j C_i^j + \sum_{i \in \mathcal{U}} q_i^j C_i^j, \\ T_\gamma &= \sum_{i \in \mathcal{L}} (1 - z_i^j) + \sum_{i \in \mathcal{U}} (1 - q_i^j), & A_\gamma &= \sum_{i \in \mathcal{L}} (1 - z_i^j) C_i^j + \sum_{i \in \mathcal{U}} (1 - q_i^j) C_i^j. \end{aligned} \tag{15}$$

Here, A_α is the expected number of correct predictions. The pairs (A_β, T_β) and (A_γ, T_γ) are the expected agreement and total counts in the correct and incorrect states. Using these counts, the parameter-dependent part of Q_j becomes:

$$\begin{aligned} Q_j = &A_\alpha \log \alpha_j + (N - A_\alpha) \log(1 - \alpha_j) \\ &+ A_\beta \log \beta_j + (T_\beta - A_\beta) \log(1 - \beta_j) \\ &+ A_\gamma \log \gamma_j + (T_\gamma - A_\gamma) \log(1 - \gamma_j). \end{aligned}$$

The three parameters are now separated. Differentiating gives:

$$\begin{aligned} \frac{\partial Q_j}{\partial \alpha_j} &= \frac{A_\alpha}{\alpha_j} - \frac{N - A_\alpha}{1 - \alpha_j}, \\ \frac{\partial Q_j}{\partial \beta_j} &= \frac{A_\beta}{\beta_j} - \frac{T_\beta - A_\beta}{1 - \beta_j}, \\ \frac{\partial Q_j}{\partial \gamma_j} &= \frac{A_\gamma}{\gamma_j} - \frac{T_\gamma - A_\gamma}{1 - \gamma_j}. \end{aligned}$$

Setting each derivative to zero and multiplying by its positive denominator gives:

$$A_\alpha(1-\alpha_j) = (N - A_\alpha)\alpha_j \quad \Longleftrightarrow \quad \alpha_j = \frac{A_\alpha}{N},$$
$$A_\beta(1-\beta_j) = (T_\beta - A_\beta)\beta_j \quad \Longleftrightarrow \quad \beta_j = \frac{A_\beta}{T_\beta},$$
$$A_\gamma(1-\gamma_j) = (T_\gamma - A_\gamma)\gamma_j \quad \Longleftrightarrow \quad \gamma_j = \frac{A_\gamma}{T_\gamma}.$$

Substituting the counts from Eq. 15 produces the three updates in Eq. 10. Finally, each scalar objective is concave because its second derivative has the following form:

$$\frac{d^2}{dp^2}\left[A\log p + (T-A)\log(1-p)\right] = -\frac{A}{p^2} - \frac{T-A}{(1-p)^2} \le 0.$$

Thus, the solution found is the global maximizer of its conditional M-step objective.

A.3 Uncertainty Reduction After Judge Validation

Entropy-guided item selection. Let $h(q) = -q\log q - (1-q)\log(1-q)$ be the binary entropy of a Bernoulli variable with success probability q. For the unjudged set $\mathcal{U}$, the posterior entropy is:

$$\mathcal{H}(\mathcal{U}) = \sum_{k\in\mathcal{U}}\sum_{j=1}^{J} h(q_k^j).$$

For a candidate item $i \in \mathcal{U}$, let $\mathcal{Y}_i = \mathcal{A}_i \cup \{\perp\}$ contain the distinct model answers and the outcome $\perp$, indicating that none is correct. For $a \neq \perp$, define the supporting models by $\mathcal{J}_i(a) = \{j : r_i^j \equiv_i a\}$. A possible answer a determines $z_i^j(a) = \mathbb{I}(r_i^j \equiv_i a)$ for $a \neq \perp$ and $z_i^j(\perp) = 0$. Write $\mathbf{z}_i(a) = (z_i^1(a), \ldots, z_i^J(a))$ for this hard correctness vector. Under the factorized-posterior approximation, the support of a is the probability that every supporting model is correct and every model $k \notin \mathcal{J}_i(a)$ is wrong:

$$\rho_i(a) = \prod_{j=1}^{J}(q_i^j)^{z_i^j(a)}(1-q_i^j)^{1-z_i^j(a)} = \prod_{j\in\mathcal{J}_i(a)} q_i^j \prod_{k\notin\mathcal{J}_i(a)}(1-q_i^k), \quad a \neq \perp,$$
$$\rho_i(\perp) = \prod_{j=1}^{J}(1-q_i^j).$$

Let $\mathbf{Z}_i = (Z_i^1, \ldots, Z_i^J)$. The events $\{\mathbf{Z}_i = \mathbf{z}_i(a) : a \in \mathcal{Y}_i\}$ are mutually exclusive. Define $\Omega_i = \{\mathbf{Z}_i = \mathbf{z}_i(a) \text{ for some } a \in \mathcal{Y}_i\}$, and let $\mathbf{C} = \{C_k^j\}_{k,j}$ be the observed agreement matrix. Therefore, normalizing gives the judge-answer distribution conditioned on Ω_i:

$$\pi_i(a) = \mathbb{P}(g_i = a \mid \mathbf{C}, \boldsymbol{\theta}, \Omega_i) = \frac{\rho_i(a)}{\rho_i(\perp) + \sum_{a'\in\mathcal{Y}_i\setminus\{\perp\}}\rho_i(a')}, \qquad a \in \mathcal{Y}_i,$$

where $\sum_{a\in\mathcal{Y}_i}\pi_i(a) = 1$. Let $q_{k,i}^{j,a}$ denote the posterior correctness of model j on each remaining item $k \in \mathcal{U}\setminus\{i\}$ after substituting $z_i^j(a)$ into the judge-conditioned EM updates. The resulting entropy is:

$$\mathcal{H}_i^a = \sum_{k\in\mathcal{U}\setminus\{i\}}\sum_{j=1}^{J} h\left(q_{k,i}^{j,a}\right).$$

Because g_i takes the value a with probability $\pi_i(a)$, the law of total expectation gives the expected entropy after validating item i:

$$\mathbb{E}[\mathcal{H} \mid \text{validate}(i)] = \sum_{a\in\mathcal{Y}_i}\pi_i(a)\mathcal{H}_i^a. \tag{16}$$

Information gain is current entropy minus expected remaining entropy. Substituting Eq. 16 gives:

$$\mathrm{IG}(i) = \mathcal{H}(\mathcal{U}) - \sum_{a \in \mathcal{Y}_i} \pi_i(a) \mathcal{H}_i^a,$$

which is the information-gain expression in Eq. 8. Since $\mathcal{H}(\mathcal{U})$ is fixed before the next item is chosen, the greatest expected reduction is obtained by the selection rule in the same equation. This proves the rule, with $i^\star$ queried only when $\mathrm{IG}(i^\star) > 0$. Validation replaces $q_{i^\star}^j$ with $z_{i^\star}^j \in \{0, 1\}$, removing its entropy since $h(0) = h(1) = 0$, while the look-ahead score captures changes to the remaining posteriors.

Parameter uncertainty. For one model–item pair, define the complete-data score vector as:

$$\mathbf{S}_{ij}(z, \boldsymbol{\theta}_j) = \nabla_{\boldsymbol{\theta}_j} \log p(C_i^j, z \mid \boldsymbol{\theta}_j), \qquad z \in \{0, 1\}.$$

Its three coordinates are the score directions for α_j, β_j, and γ_j. Define the difference between its two possible values as:

$$\mathbf{d}_i^j = \mathbf{S}_{ij}(1, \boldsymbol{\theta}_j) - \mathbf{S}_{ij}(0, \boldsymbol{\theta}_j).$$

Conditioned on C_i^j, the latent variable Z_i^j is Bernoulli with probability q_i^j. Therefore, the conditional score covariance is:

$$\mathrm{Cov}(\mathbf{S}_{ij}(Z_i^j, \boldsymbol{\theta}_j) \mid C_i^j) = q_i^j (1 - q_i^j) \mathbf{d}_i^j (\mathbf{d}_i^j)^\mathsf{T} \succeq \mathbf{0}.$$

After exact validation, Z_i^j is known and the same covariance is $\mathrm{Cov}(\mathbf{S}_{ij}(Z_i^j, \boldsymbol{\theta}_j) \mid C_i^j, Z_i^j) = \mathbf{0}$. Hence, for any exactly validated set $\mathcal{S} \subseteq \{1, \ldots, N\}$, the total remaining score uncertainty is:

$$\mathcal{V}_j(\mathcal{S}) = \sum_{i \notin \mathcal{S}} \mathbb{E}_{C_i^j} \left[\mathrm{Cov}(\mathbf{S}_{ij}(Z_i^j, \boldsymbol{\theta}_j) \mid C_i^j) \right].$$

In particular, $\mathcal{V}_j(\varnothing)$ sums over all N target items, while $\mathcal{V}_j(\mathcal{L})$ sums only over the unjudged set $\mathcal{U}$. Subtracting the two matrices gives:

$$\mathcal{V}_j(\varnothing) - \mathcal{V}_j(\mathcal{L}) = \sum_{i \in \mathcal{L}} \mathbb{E}_{C_i^j} \left[q_i^j (1 - q_i^j) \mathbf{d}_i^j (\mathbf{d}_i^j)^\mathsf{T} \right] \succeq \mathbf{0}.$$

Because $A \preceq B$ means that $B - A$ is positive semidefinite, the preceding result is equivalent to:

$$\boxed{\mathcal{V}_j(\mathcal{L}) \preceq \mathcal{V}_j(\varnothing).}$$

This proves that correct validation cannot increase score uncertainty in any parameter direction. For interior parameters, the α_j coordinate of $\mathbf{d}_i^j$ is $1/[\alpha_j(1 - \alpha_j)] > 0$, so $\mathbf{d}_i^j \neq \mathbf{0}$. The inequality is strict in at least one direction whenever some validated pair had $0 < q_i^j < 1$ before validation.

A.4 Warm Starts After Judge Validation

We now derive the warm-start bounds, which quantify how far judge validation moves the solution and justify reusing the previously converged parameters. Suppress j and define the expected counts before validating item k as $A = \sum_i q_i$, $B = \sum_i q_i C_i$, $W = \sum_i (1 - q_i)$, and $H = \sum_i (1 - q_i) C_i$. The current M-step gives $\bar{\alpha} = A/N$, $\bar{\beta} = B/A$, and $\bar{\gamma} = H/W$. Let $\delta_k = z_k - q_k$. Replacing q_k by z_k gives the exact changes:

$$\alpha' - \bar{\alpha} = \frac{\delta_k}{N}, \qquad \beta' - \bar{\beta} = \frac{\delta_k (C_k - \bar{\beta})}{A + \delta_k}, \qquad \gamma' - \bar{\gamma} = \frac{\delta_k (\bar{\gamma} - C_k)}{W - \delta_k}.$$

If $A + \delta_k \geq cN$ and $W - \delta_k \geq cN$ for $c > 0$, then $|\delta_k| \leq 1$ and all probabilities lie in $[0, 1]$:

$$\left\| \begin{bmatrix} \alpha' - \bar{\alpha} \\ \beta' - \bar{\beta} \\ \gamma' - \bar{\gamma} \end{bmatrix} \right\|_2 \leq \frac{1}{N} \sqrt{1 + \frac{2}{c^2}} = O\left(\frac{1}{N}\right).$$

To bound the converged solution, let v index judge rounds, with validated and unjudged sets $\mathcal{L}_v$ and $\mathcal{U}_v$. Define the judge-conditioned log-likelihood and its normalized objective by:

$$\ell_v(\boldsymbol{\theta}) = \sum_{j=1}^{J} \left[\sum_{i \in \mathcal{L}_v} \log p(C_i^j, z_i^j \mid \boldsymbol{\theta}_j) + \sum_{i \in \mathcal{U}_v} \log \sum_{z \in \{0,1\}} p(C_i^j, z \mid \boldsymbol{\theta}_j) \right],$$
$$F_v(\boldsymbol{\theta}) = \frac{1}{N} \ell_v(\boldsymbol{\theta}).$$

If round $v+1$ validates item $k \in \mathcal{U}_v$, define its change to the unnormalized objective as:

$$\Delta_k(\boldsymbol{\theta}) = \sum_{j=1}^{J} \left[\log p(C_k^j, z_k^j \mid \boldsymbol{\theta}_j) - \log \sum_{z \in \{0,1\}} p(C_k^j, z \mid \boldsymbol{\theta}_j) \right].$$

Therefore, $F_{v+1} = F_v + N^{-1}\Delta_k$. Let $\widehat{\boldsymbol{\theta}}_v$ and $\widehat{\boldsymbol{\theta}}_{v+1}$ be local maximizers of F_v and F_{v+1}. Assume that both lie in a shared neighborhood where F_{v+1} is μ-strongly concave for some $\mu > 0$, and let G bound the perturbation gradient there, so $\|\nabla \Delta_k(\boldsymbol{\theta})\|_2 \le G$. Strong concavity and Cauchy–Schwarz then give:

$$\mu \left\| \widehat{\boldsymbol{\theta}}_{v+1} - \widehat{\boldsymbol{\theta}}_v \right\|_2 \le \left\| \nabla F_{v+1}(\widehat{\boldsymbol{\theta}}_v) \right\|_2.$$

Because $\nabla F_v(\widehat{\boldsymbol{\theta}}_v) = \mathbf{0}$ and $\|\nabla \Delta_k(\widehat{\boldsymbol{\theta}}_v)\|_2 \le G$, norm homogeneity gives:

$$\left\| \nabla F_{v+1}(\widehat{\boldsymbol{\theta}}_v) \right\|_2 = \frac{1}{N} \left\| \nabla \Delta_k(\widehat{\boldsymbol{\theta}}_v) \right\|_2 \le \frac{G}{N}.$$

Combining the last two inequalities gives the warm-start bound:

$$\left\| \widehat{\boldsymbol{\theta}}_{v+1} - \widehat{\boldsymbol{\theta}}_v \right\|_2 \le \frac{G}{\mu N} = O\left(\frac{1}{N}\right).$$

Thus, the previously converged parameters provide an $O(1/N)$ warm start for the updated local objective under the stated local assumptions.

B EX AND EX-EXT

Standard execution accuracy (EX) compares the predicted and reference queries on the original database D shipped with the benchmark, so two queries that mean different things match whenever that database cannot tell them apart. EX-Ext instead compares their results across modified database instances. Tab. 2 illustrates insertions, an update, and the removal of one or more rows. GPT-5.4 guided the selection of inserted rows to expose differences between semantically distinct queries.

Original database (D)

name	dept	salary
Alice	Research	52000
Bob	Research	58000
Carol	Sales	71000
Dan	Marketing	64000

Instance 1

name	dept	salary
Alice	Research	52000
Bob	Research	58000
Carol	Sales	71000
Dan	Marketing	64000
Erin	Research	83000

Instance 2

name	dept	salary
Alice	Research	52000
Bob	Research	58000
Carol	Sales	71000
Dan	Marketing	64000
Frank	Sales	55000

Instance 3

name	dept	salary
Alice	Research	52000
Bob	Research	61000
Carol	Sales	71000
Dan	Marketing	64000

Instance 4

name	dept	salary
Bob	Research	58000
Carol	Sales	71000
Dan	Marketing	64000

Removed: Alice

Instance 5

name	dept	salary
Carol	Sales	71000
Dan	Marketing	64000

Removed: Alice, Bob

Question	Which employees work in the Research department?
Reference	`SELECT name FROM employee WHERE dept = 'Research';`
Prediction	`SELECT name FROM employee WHERE salary BETWEEN 50000 AND 60000;`

	Original D	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
Reference returns	Alice, Bob	Alice, Bob, Erin	Alice, Bob	Alice, Bob	Bob	∅
Prediction returns	Alice, Bob	Alice, Bob	Alice, Bob, Frank	Alice	Bob	∅
Results agree	✓	✗	✗	✗	✓	✓

Table 2: Original database and modified instances used to compare EX and EX-Ext.

On the original database D, both queries return Alice and Bob, so EX accepts the incorrect prediction. Instance 1 adds a Research employee outside the predicted salary range, Instance 2 adds a non-Research employee inside that range, and Instance 3 moves Bob outside the range. These changes expose three different disagreements. Instances 4 and 5 remove one and two rows, respectively. They preserve agreement because the two predicates already agree on every row in D, so deleting those rows cannot separate the queries. EX-Ext rejects the prediction because agreement must hold across every instance, and the failures on Instances 1 to 3 are sufficient.

C TEXT2SQL MODEL POOL

We form a fixed pool of $H = 35$ models from Hugging Face, listed in Tab. 3. We exclude quantized mirrors and duplicate checkpoints. Parameter counts come from the corresponding model pages, with a dash denoting an unreported count.

D IMAGE CLASSIFICATION

Datasets and Baselines. We evaluate on USPS (Hull, 1994), SVHN (Netzer et al., 2011), CIFAR-10.1 (Recht et al., 2018), CIFAR-10-C (Hendrycks & Dietterich, 2019), ImageNet-V2 (Recht et al., 2019), ImageNet-R (Hendrycks et al., 2021a), and ImageNet-Sketch (Wang et al., 2019), using MNIST, CIFAR-10, and ImageNet as the matching sources. Each target uses a compatible pool and class mapping, with labels hidden from all evaluators. Baselines include AutoEval (Deng & Zheng, 2021), MetaEvaluator (Pham et al., 2026a), GDE (Jiang et al., 2022), ALine-D (Baek et al., 2022), Discrepancy (Rosenfeld & Garg, 2023), ATC (Garg et al., 2022), DoC (Guillory et al., 2021), GLAD (Whitehill et al., 2009), DawidSkene (Dawid & Skene, 1979), SpectralDS (Zhang et al., 2016), and LearnCrowds (Raykar et al., 2010). Crowd aggregators treat models as annotators and estimate model j's accuracy by averaging $q_i(r_i^j)$. SQL-specific methods and ConfProfile (Fu et al., 2023) do not apply.

Model Pools. The pool holds 20 backbones, listed in Tab. 4: ten from the architecture families used by MetaEvaluator (Pham et al., 2026a), six widely downloaded Hugging Face classification

Model	#Params	Link
Arctic-Text2SQL-R1-7B	7.6B	Snowflake/Arctic-Text2SQL-R1-7B
Code-Llama-2-13B-Instruct-Text2SQL	13.0B	bugdaryan/Code-Llama-2-13B-instruct-text2sql
CodeLlama-70B-Instruct	69.0B	codellama/CodeLlama-70b-Instruct-hf
CodeS-1B	1.0B	seeklhy/codes-1b
CodeS-3B	3.0B	seeklhy/codes-3b
CodeS-7B	7.0B	seeklhy/codes-7b
CodeS-15B	15.0B	seeklhy/codes-15b
CscSQL-Merge-Qwen2.5-Coder-7B	7.6B	cycloneboy/CscSQL-Merge-Qwen2.5-Coder-7B-Instruct
Distil-Qwen3-4B-Text2SQL	4.0B	distil-labs/distil-qwen3-4b-text2sql
Flan-T5-Text2SQL-Schema	247.6M	juierror/flan-t5-text2sql-with-schema
Flan-T5-Text2SQL-Schema-v2	247.5M	juierror/flan-t5-text2sql-with-schema-v2
Gemma-2-2B	2.6B	google/gemma-2-2b
Graphix-3B	3.0B	patrickNLP/Graphix-3B
Hrida-T2SQL-3B	3.8B	HridaAI/Hrida-T2SQL-3B-V0.2
Llama-3-SQLCoder-8B	8.0B	defog/llama-3-sqlcoder-8b
Llama-3.1-70B-Instruct	70.6B	meta-llama/Llama-3.1-70B-Instruct
Meta-Llama-3-70B	70.6B	meta-llama/Meta-Llama-3-70B
Ministral-3-14B	13.9B	mistralai/Ministral-3-14B-Instruct-2512
Mixtral-8x7B-Instruct	46.7B	mistralai/Mixtral-8x7B-Instruct-v0.1
OmniSQL-7B	7.6B	seeklhy/OmniSQL-7B
OmniSQL-14B	14.8B	seeklhy/OmniSQL-14B
Pip-SQL-1.3B	1.3B	PipableAI/pip-sql-1.3b
Qwen2.5-32B	32.8B	Qwen/Qwen2.5-32B
Qwen2.5-72B-Instruct	72.7B	Qwen/Qwen2.5-72B-Instruct
Qwen3-4B-Text-to-SQL	4.0B	Ellbendls/Qwen-3-4b-Text_to_SQL
SQLCoder-7B-2	6.7B	defog/sqlcoder-7b-2
SQLCoder-15B	15.5B†	defog/sqlcoder
SQLCoder-70B-Alpha	69.0B	defog/sqlcoder-70b-alpha
SQLCoder2	15.5B†	defog/sqlcoder2
T5-LM-Large-Text2SQL-Spider	783.2M	gaussalgo/T5-LM-Large-text2sql-spider
T5-Small-Text-to-SQL	60.5M†	cssupport/t5-small-awesome-text-to-sql
XiYanSQL-QwenCoder-3B	3.4B	XGenerationLab/XiYanSQL-QwenCoder-3B-2502
XiYanSQL-QwenCoder-7B	7.6B	XGenerationLab/XiYanSQL-QwenCoder-7B-2502
XiYanSQL-QwenCoder-14B	14.8B	XGenerationLab/XiYanSQL-QwenCoder-14B-2502
XiYanSQL-QwenCoder-32B	32.8B	XGenerationLab/XiYanSQL-QwenCoder-32B-2504

Table 3: Text2SQL model pool. Parameter counts are read from the Hugging Face model pages. Counts marked † are not reported on the model page and are derived from the sizes of the published weight files and their stored precision.

checkpoints, and four vision–language models (Radford et al., 2021; Zhai et al., 2023) that classify zero-shot by scoring the target class names. Every model predicts into the target class vocabulary, with classification heads matched to each source and target rather than shared across the digit, CIFAR, and ImageNet tasks.

Metrics and Implementation Details. True performance is top-1 classification accuracy, the percentage of target images assigned the correct class, so the execution-based EX-Ext metric of §5.1 does not apply here. We report MAE against that accuracy in percentage points. The judge is Qwen2.5-VL-72B-Instruct[1], a vision–language model chosen because no backbone in Tab. 4 belongs to its family or shares its vision encoder, so its verdicts stay independent of the pool it validates. The budgets are selected from Fig. 11, which gives $V = 6$ single-item judge rounds and $K = 10$ meta-dataset subsets.

Performance Comparison. Tab. 5 shows that PoolEvaluator achieves the lowest MAE across all image targets. Fig. 8 further shows that its estimates remain closest to ground-truth accuracy under digit, corruption, and rendition shifts. PoolEvaluator underestimates accuracy on the easier USPS and SVHN targets and overestimates it on the harder CIFAR-10-C and ImageNet-R. This matches the Text2SQL pattern in §5.2.

[1] https://huggingface.co/Qwen/Qwen2.5-VL-72B-Instruct

Model	#Params	Link
CLIP-ViT-B/32	151.3M†	openai/clip-vit-base-patch32
CLIP-ViT-L/14	427.6M	openai/clip-vit-large-patch14
ConvNeXt-Tiny	28.6M	timm/convnext_tiny.fb_in1k
ConvNeXtV2-Base	88.7M	facebook/convnextv2-base-22k-384
DeiT-Small	22.1M	timm/deit_small_patch16_224.fb_in1k
DenseNet-121	8.1M	timm/densenet121.ra_in1k
EfficientNet-B2	9.2M	timm/efficientnet_b2.ra_in1k
MobileNetV2	3.5M	timm/mobilenetv2_100.ra_in1k
MobileNetV3-Small	2.6M	timm/mobilenetv3_small_100.lamb_in1k
RegNetY-8GF	39.3M	timm/regnety_080.ra3_in1k
ResNet-18 – Microsoft	11.7M	microsoft/resnet-18
ResNet-18 – timm	11.7M	timm/resnet18.a1_in1k
ResNet-50 – Microsoft	25.6M	microsoft/resnet-50
ResNeXt-50-32x4d	25.1M	timm/resnext50_32x4d.a1_in1k
SigLIP2-Base/16	375.2M	google/siglip2-base-patch16-224
SigLIP-SO400M/14	878.0M	google/siglip-so400m-patch14-384
VGG-16	138.4M	timm/vgg16.tv_in1k
ViT-Base	86.6M	google/vit-base-patch16-224
ViT-Tiny	5.7M	timm/vit_tiny_patch16_224.augreg_in21k_ft_in1k
WideResNet-50-2	69.0M	timm/wide_resnet50_2.racm_in1k

Table 4: Image classification model pool. Parameter counts are read from the Hugging Face model pages, and counts marked † are derived instead from the published weight files.

Table 5: MAE in percentage points (mean ± 95% CI, lower is better). Each column is a target dataset. Avg. is the mean across the seven datasets, with the CI averaged over columns. Best results are **bold** and second-best results are underlined.

Method	USPS	SVHN	CIFAR-10.1	CIFAR-10-C	ImageNet-V2	ImageNet-R	ImageNet-Sketch	Avg.
ATC	8.83±1.54	12.67±1.92	10.21±1.88	12.31±2.06	11.78±1.84	13.53±2.60	13.55±2.51	11.84±2.05
DoC	8.61±1.26	12.22±1.82	8.95±1.45	12.26±1.55	10.60±1.43	12.59±1.75	13.66±2.08	11.27±1.62
GDE	7.98±1.71	10.13±2.40	9.72±2.14	10.46±2.05	10.08±2.27	12.89±2.84	13.08±2.27	10.62±2.24
AutoEval	7.92±1.28	10.65±1.58	8.05±1.11	10.68±1.60	9.43±1.27	10.98±1.66	11.94±1.86	9.95±1.48
ALine-D	6.73±1.39	9.53±2.20	7.96±1.64	9.54±1.89	8.62±1.72	10.39±2.39	12.40±2.28	9.31±1.93
DawidSkene	6.43±1.02	8.78±1.42	7.80±1.15	9.06±1.40	8.41±1.12	10.87±1.56	9.97±1.50	8.76±1.31
LearnCrowds	6.23±1.42	8.82±1.60	6.34±1.68	8.20±1.95	8.06±1.49	10.05±1.88	9.21±2.37	8.13±1.77
SpectralDS	6.08±0.92	7.97±1.12	6.61±0.92	7.15±1.28	7.63±1.07	9.30±1.15	8.74±1.38	7.64±1.12
GLAD	5.54±1.24	7.30±1.62	6.12±1.28	7.87±1.53	7.21±1.36	7.78±1.86	9.14±1.96	7.28±1.55
Discrepancy	5.26±0.75	6.76±0.92	5.78±0.81	6.94±0.94	7.13±0.97	8.12±1.09	9.15±1.31	7.02±0.97
MetaEvaluator	3.67±1.06	5.24±1.44	4.28±1.23	5.04±1.42	4.58±1.39	5.67±1.74	6.31±1.94	4.97±1.46
PoolEvaluator	**2.36±0.62**	**3.06±0.86**	**2.62±0.76**	**3.61±0.98**	**3.00±0.92**	**3.76±1.12**	**3.64±0.97**	**3.15±0.89**

Ranking Stability. Fig. 9 shows that the lower estimation error yields stronger pairwise, Kendall, Top-1, and Top-5 ranking quality, and that Kendall's τ stays high with the narrowest interval as the pool grows to 20 backbones, while the baselines degrade.

Evaluation Efficiency. Fig. 10 shows that PoolEvaluator maintains the lowest MAE as the pool grows. It also avoids evaluator training, keeping total latency below the baselines.

Budget Selection. Fig. 11 reports the two sweeps behind these settings. Warm-starting EM from the previous solution saves about 44% of the iterations at $V = 6$. Both the iteration reduction and MAE improve slowly beyond that point. MAE likewise plateaus by $K = 10$ subsets. Even with a random prior, PoolEvaluator reaches a lower MAE than MetaEvaluator.

E Node Classification

Datasets and Baselines. We evaluate on ACMv9, Citationv1, and DBLPv7 (Zheng et al., 2023), ogbn-arxiv (Hu et al., 2020), and GOOD-Cora, GOOD-Twitch, and GOOD-WebKB (Gui et al.,

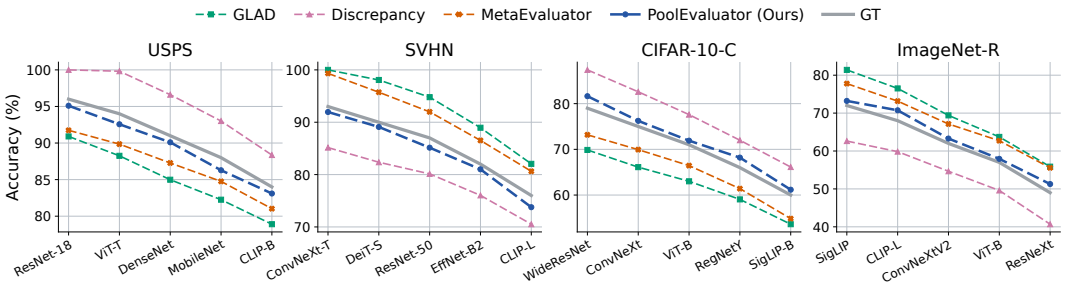


Figure 8: Estimated and ground-truth image classification accuracy across representative targets. Curves closer to ground truth indicate more accurate estimates.

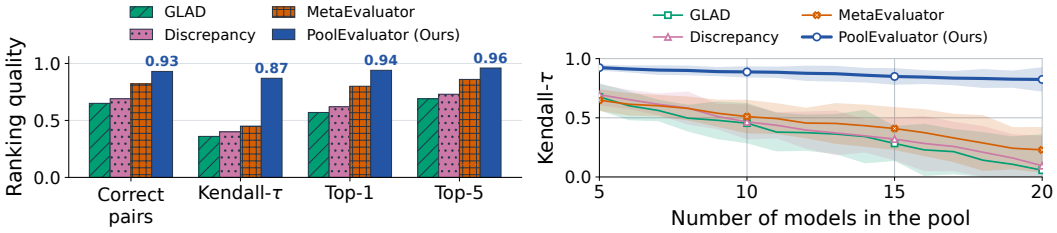


Figure 9: *Left*: ranking quality averaged over pools of 5 to 20 models. Higher is better for all four metrics. *Right*: Kendall's τ against pool size, with shaded 95% CIs over repeated runs.

2022), which cover citation, social, and web-page graphs under the predefined GOOD shift settings. Source and target splits preserve each target's class vocabulary, and target-node labels stay hidden during evaluation. We compare PoolEvaluator with GNNEvaluator (Zheng et al., 2023), MetaEvaluator (Pham et al., 2026a), ALine-D (Baek et al., 2022), GDE (Jiang et al., 2022), Majority, DawidSkene (Dawid & Skene, 1979), SpectralDS (Zhang et al., 2016), and LearnCrowds (Raykar et al., 2010). GDE runs within each architecture family across its three seeds.

Model Pools. Each target draws on a compatible pool of 20 models listed in Tab. 6, collected through a broad search of relevant node classification architectures and released graph checkpoints. The pool comprises 16 node classification architectures, 1 graph foundation model, and 3 graph language models (Chen et al., 2024b; Tang et al., 2024) that answer node queries in text. The graph language models provide architectural diversity, and their generated answers are mapped to the target class vocabulary before agreement is computed. Architectures are trained per target with three seeds each, with labeled source nodes available wherever a baseline needs them for calibration.

Metrics and Implementation Details. True performance is node classification accuracy, the percentage of target nodes assigned the correct class, so EX-Ext does not apply here. We report MAE against that accuracy in percentage points, and total latency includes evaluator fitting, adaptation, and full target-graph inference when required. The judge is GOFA[2], a generative graph foundation model that answers node classification queries over text-attributed graphs. Its Mistral backbone and its own family are absent from Tab. 6, so the judge shares no similarity or bias with the models it validates. The budgets are selected from Fig. 15, which gives $V = 5$ single-item judge rounds and $K = 12$ subsets.

Performance Comparison. Tab. 7 shows that PoolEvaluator achieves the lowest MAE across all graph targets. Fig. 12 confirms that its estimates track ground-truth accuracy more closely under structural and domain shifts. Its deviations also follow a consistent direction within each target. PoolEvaluator underestimates accuracy on the easier ACMv9 and DBLPv7 graphs and overestimates it on the harder ogbn-arxiv and GOOD-WebKB, matching the Text2SQL pattern reported in §5.2.

Ranking Stability. Fig. 13 shows consistent gains in pairwise ordering. Kendall's τ also holds up with the narrowest interval as the compatible pool grows to 20 models.

[2] https://github.com/JiaruiFeng/GOFA

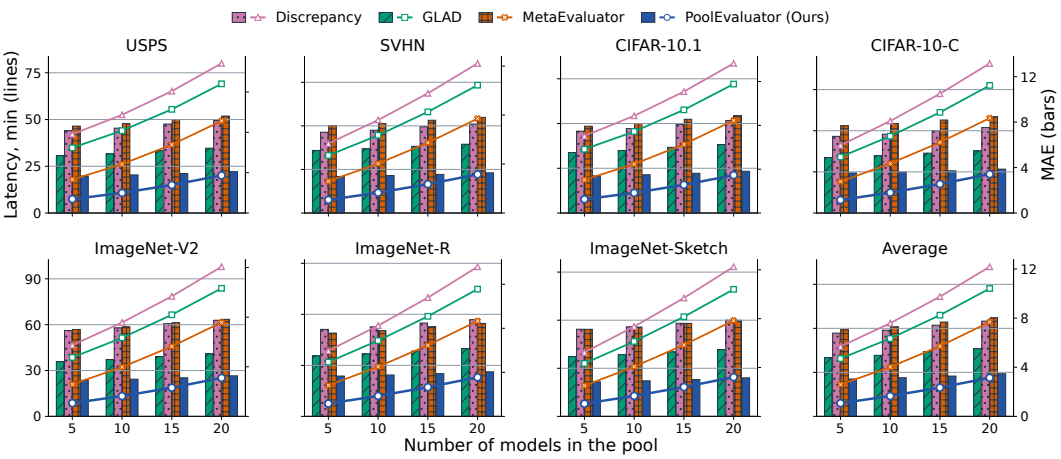


Figure 10: MAE and total evaluation latency across model-pool sizes. Lines use the left axis and bars use the right axis. Lower is better.

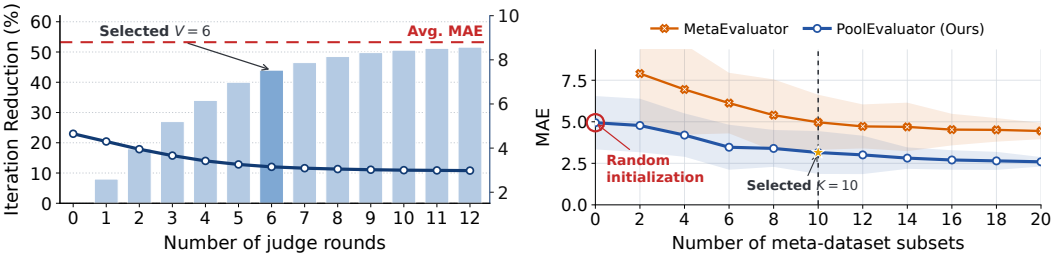


Figure 11: *Left*: iteration reduction (bars, left axis) and MAE (blue line, right axis) across single-item judge rounds. *Right*: MAE by the number of selected meta-dataset subsets, where $K = 0$ denotes random prior initialization. Shading denotes 95% CIs. The red line marks the average MAE of the baseline methods in Tab. 5.

Evaluation Efficiency. Fig. 14 shows that PoolEvaluator retains the lowest MAE and latency as the graph-model pool grows. GNNEvaluator and MetaEvaluator incur evaluator fitting or adaptation, while PoolEvaluator reuses shared target predictions.

Budget Selection. Fig. 15 shows that iteration reduction reaches about 41% by $V = 5$, with little gain from additional judge rounds, while MAE plateaus at $K = 12$ subsets. Even with random initialization, PoolEvaluator achieves lower MAE than GNNEvaluator and MetaEvaluator at their smallest subset budgets, reducing reliance on costly labeled graph meta-subsets.

Model	#Params	Link
APPNP	–	PyG APPNP
ARMA	–	PyG ARMAConv
ChebNet	–	PyG ChebConv
GAT	–	PyG GATConv
GATv2	–	PyG GATv2Conv
GCN	–	PyG GCNConv
GCNII	–	PyG GCN2Conv
GIN	–	PyG GINConv
GraphGPT-7B	6.9B†	Jiabin99/GraphGPT-7B-mix-all
GraphPFN-1.3	3.6M‡	eremeev-d/graphpfn-1.3
GraphSAGE	–	PyG SAGEConv
JKNet	–	PyG JumpingKnowledge
LLaGA-HO-Classification	6.7B‡	Runjin/llaga-vicuna-7b-simteg-HO-classification_expert-linear-projector
LLaGA-ND-Classification	6.7B‡	Runjin/llaga-vicuna-7b-simteg-ND-classification_expert-linear-projector
MixHop	–	PyG MixHopConv
MLP	–	PyG MLP
SGC	–	PyG SGConv
SuperGAT	–	PyG SuperGATConv
TAGCN	–	PyG TAGConv
UniMP	–	PyG TransformerConv

Table 6: Sixteen architectures are instantiated per target graph, so their parameter counts vary with the input dimension and the number of classes and are left unset. Counts marked † are derived from the published weight files and their stored precision. Models marked ‡ provide only adapter or projector weights. LLaGA is counted with its required Vicuna-7B-v1.5-16k base and GraphPFN by the adapter alone.

Table 7: MAE in percentage points (mean ± 95% CI, lower is better). Each column is a target dataset. Avg. is the mean across the seven datasets, with the CI averaged over columns. Best results are **bold** and second-best results are underlined.

Method	ACMv9	Citationv1	DBLPv7	ogbn-arxiv	GOOD-Cora	GOOD-Twitch	GOOD-WebKB	Avg.
DawidSkene	12.42±2.16	12.04±1.93	13.85±2.30	16.09±2.30	13.80±2.40	16.45±2.96	17.69±2.82	14.62±2.41
Majority	10.55±1.50	11.02±1.55	11.90±1.44	14.02±1.91	13.93±1.70	16.35±1.99	16.38±2.37	13.45±1.78
SpectralDS	9.67±2.11	10.60±1.73	11.98±2.08	13.42±2.05	12.10±2.21	13.52±2.22	13.97±2.44	12.18±2.12
ALine-D	8.91±1.03	9.88±1.28	10.27±1.48	10.75±1.53	11.47±1.53	12.25±1.95	12.49±1.98	10.86±1.54
LearnCrowds	7.90±1.49	7.86±1.55	9.70±1.83	10.09±2.08	9.97±1.94	11.27±2.23	11.39±2.60	9.74±1.96
GDE	6.24±1.19	7.05±1.26	8.42±1.10	8.84±1.57	7.72±1.32	9.93±1.34	9.97±1.53	8.31±1.33
GNNEvaluator	4.95±1.39	5.53±1.29	6.45±1.75	6.59±1.74	6.00±1.45	7.56±2.12	7.86±2.02	6.42±1.68
MetaEvaluator	4.36±0.90	4.03±0.93	4.50±1.09	4.80±1.36	4.80±1.03	5.90±1.18	5.98±1.63	4.91±1.16
PoolEvaluator	**2.01±0.43**	**2.12±0.40**	**2.50±0.50**	**2.98±0.60**	**2.61±0.52**	**2.84±0.61**	**3.00±0.65**	**2.58±0.53**

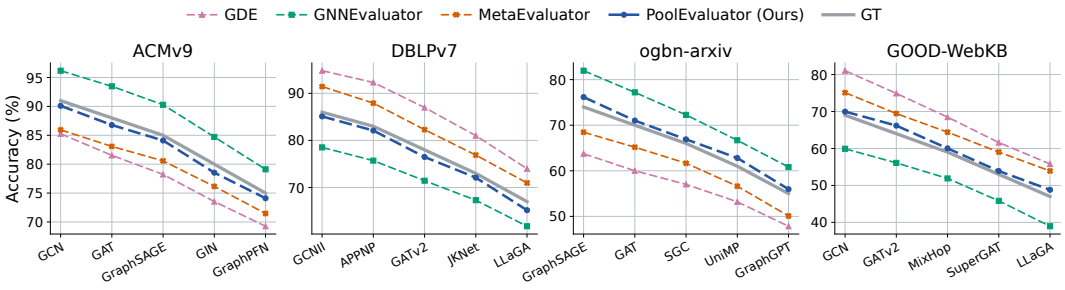


Figure 12: Estimated and ground-truth node classification accuracy across representative graph targets. Curves closer to ground truth indicate more accurate estimates.

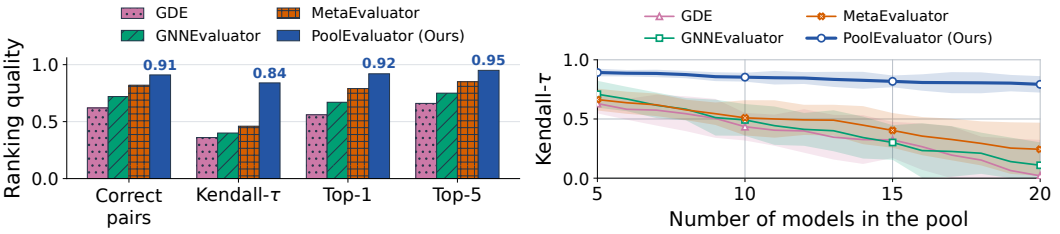


Figure 13: *Left*: ranking quality averaged over pools of 5 to 20 models. Higher is better for all four metrics. *Right*: Kendall's τ against pool size, with shaded 95% CIs over repeated runs.

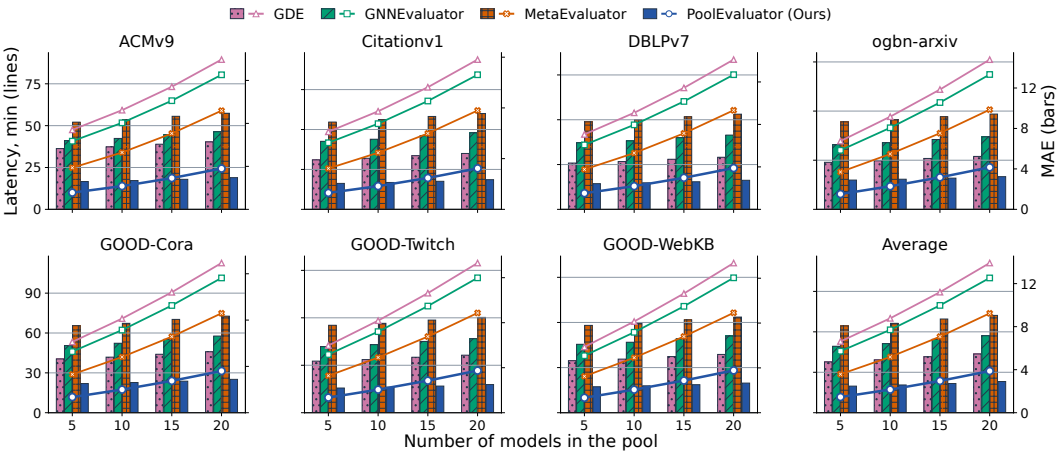


Figure 14: MAE and total evaluation latency across model-pool sizes. Lines use the left axis and bars use the right axis. Lower is better.

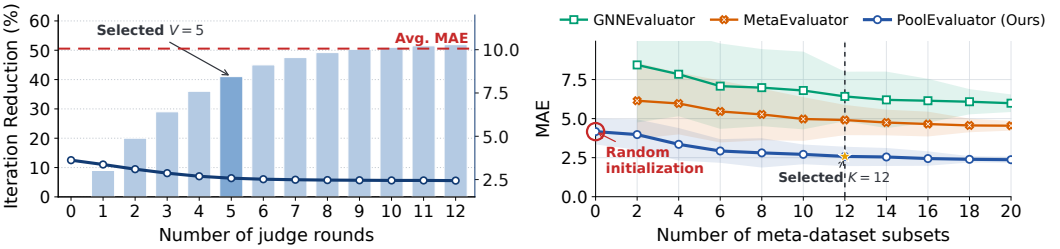


Figure 15: *Left*: iteration reduction (bars, left axis) and MAE (blue line, right axis) across single-item judge rounds. *Right*: MAE by the number of selected meta-dataset subsets, where $K = 0$ denotes random prior initialization. Shading denotes 95% CIs. The red line marks the average MAE of the baseline methods in Tab. 7.