MODELLENS: Finding the Best for Your Task from Myriads of Models

Rui Cai[†], **Weijie Jacky Mo**[†], **Xiaofei Wen**[†], **Qiyao Ma**[†],
Wenhui Zhu[‡], **Xiwen Chen**[§], **Muhao Chen**[†], **Zhe Zhao**[†]
[†]University of California, Davis [‡]Arizona State University [§]Morgan Stanley
ruicai@ucdavis.edu

Github: https://github.com/luisrui/ModelLens.git

Demo: huggingface.co/spaces/luisrui/ModelLens

Abstract

The open-source model ecosystem now contains hundreds of thousands of pretrained models, yet picking the best model for a new dataset is increasingly infeasible: new models and unbenchmarked datasets emerge continuously, leaving practitioners with no prior records on either side. Existing approaches handle only fragments of this *in the wild* setting: AutoML and transferability estimation select models from small predefined pools or require expensive per-model forward passes on the target dataset, while model routing presupposes a given candidate pool. We introduce MODELLENS, a unified framework for model recommendation in the wild. Our key insight is that public leaderboard interactions, though scattered and noisy, collectively trace out an implicit atlas of model capabilities across heterogeneous evaluation settings, a signal rich enough to learn from directly. By learning a performance-aware latent space over model–dataset–metric tuples, MODELLENS ranks unseen models on unseen datasets without running candidates on the target dataset. On a new benchmark of 1.62M evaluation records spanning 47K models and 9.6K datasets, MODELLENS surpasses baselines that either rely on metadata alone or require running each candidate on the target dataset. Its recommended Top-K pools further improve multiple representative routing methods by up to 81% relative EM gain across diverse QA benchmarks. Case studies on recently released benchmarks further confirm generalization to both text and vision-language tasks.

1 Introduction

The rapid growth of open-source machine learning models has created an unprecedented opportunity for practitioners to build, customize, and deploy AI systems [1, 2]. Platforms such as HuggingFace [3] now host hundreds of thousands of models spanning diverse architectures, scales, and application domains. Faced with a new task or dataset, practitioners must decide which model to adopt or fine-tune for their specific use case. Despite its importance, this decision remains notoriously difficult, and typically demands extensive empirical evaluation or ad-hoc trial-and-error [4, 5]. In this work, we take a step toward **model recommendation in the wild**, a setting in which thousands of heterogeneous models and datasets coexist across diverse architectures, modalities, and evaluation protocols.

However, existing approaches to model selection are ill-equipped for this in-the-wild setting. Automated machine learning (AutoML) methods [7, 8, 9] search over a fixed pool of models or pipelines to find the best fit for a target task. Transferability estimation [10, 11, 12] ranks pretrained models for a given dataset, typically by extracting feature or label statistics from a forward pass on the target. Model routing [13, 14, 15] performs instance-level selection over a predefined candidate pool, dispatching each query to one of a few pre-curated models. While each approach makes progress on a slice of the problem, none addresses the requirements of open model ecosystems along three axes.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

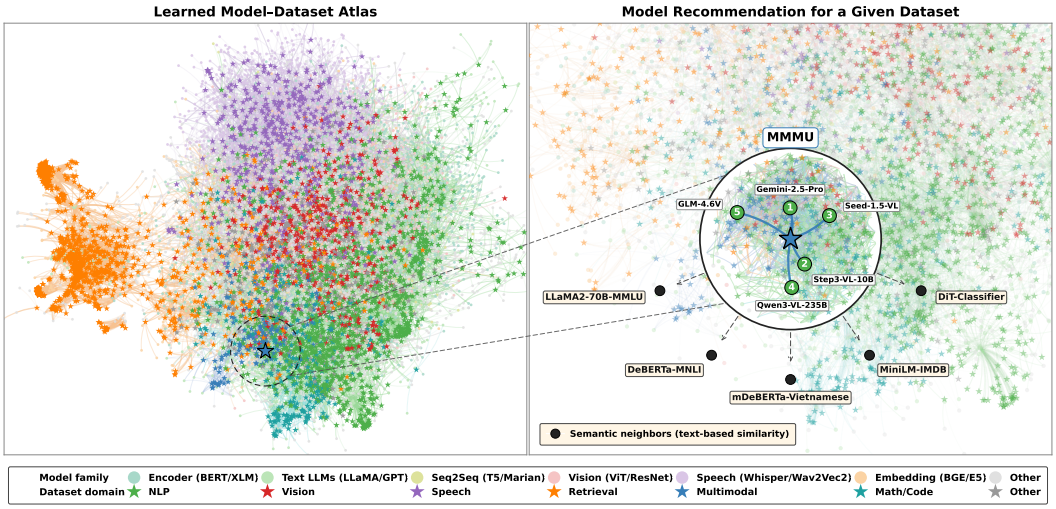

Figure 1: Model recommendation in the wild. **(Left)** Inspired by Model Atlas [6], which maps the model ecosystem by lineage, we draw a 2D map of ∼47K models (dots) and ∼9.6K datasets (⋆), positioned by embeddings learned from model–dataset performance records. The dashed circle marks the example dataset `MMMU`. **(Right)** Zoom-in around `MMMU`: the top-5 models retrieved in this learned space (green numbered badges) are competitive multimodal LLMs. In contrast, the nearest neighbours by description-embedding similarity alone (**black filled circles**, dashed arrows) are topically related but unsuitable for `MMMU` (e.g., DeBERTa-MNLI, DiT-Classifier), and lie far from it in the learned map.

Scale. AutoML and routing presuppose a small, curated pool, ignoring the hundreds of thousands of models available today; Transferability estimation is pool-agnostic but requires a forward pass per candidate, infeasible at this scale. *Generalization.* Transferability and routing methods require evaluating each candidate on the target dataset, preventing extension to newly released models and unseen datasets. *Heterogeneity.* All three lines of work assume homogeneous evaluation with a single metric on a single task family. Real benchmarks are heterogeneous even within a task family: captioning admits BLEU, ROUGE, CIDEr, and METEOR; classification admits accuracy, F1, and top-k accuracy. These metrics can rank the same model differently, so single-metric conclusions are fragile. These limitations raise a key question: *can we leverage large-scale model–dataset interaction patterns to enable model selection in the wild, without requiring direct evaluation or fine-tuning?*

Our key insight is that the seemingly fragmented, large-scale interactions between models and datasets on modern leaderboards are not merely noise but a rich source of implicit supervision, encoding how model capabilities align with dataset characteristics. Figure 1 illustrates this on a real subset of our data: when models and datasets are projected into a space learned from interactions, they cluster naturally by modality and task type, whereas a space induced from textual descriptions alone fails to recover this structure (Figure 5). For a target benchmark such as MMMU [16], the learned space surfaces the real competitive multimodal LLMs as nearest neighbors, while raw description similarity retrieves semantically related but performance-irrelevant models (e.g., DeBERTa-MNLI). This motivates formulating model recommendation as a learning problem over model–dataset interactions, providing recommendations without ever running candidate models on the target dataset. We instantiate this idea by aggregating performance records from public leaderboards [1, 2, 3] into a unified repository, with each entry represented as a tuple (*model, dataset, metric, performance*), and casting model recommendation in the wild as a ranking problem over these interactions.

Broadly, MODELLENS takes target dataset and candidate model descriptions together with leaderboard interactions as input, and outputs a ranking of candidates by predicted performance. Recommended models can be deployed via any downstream pipeline, such as zero-shot inference, in-context learning, fine-tuning, or routing. Specifically, MODELLENS introduces a structural prior over model scale and architecture family to capture predictable trends like neural scaling, paired with a learned interaction term for fine-grained model–dataset compatibility. To support cold-start inference on newly released models and unbenchmarked datasets, each entity is represented by identity, family, name, and description embeddings, with ID-dropout applied during training to force reliance on

metadata when identity is unavailable. We validate MODELLENS on 1.62M evaluation records spanning 47K models and 9.6K datasets, across matrix completion, held-out datasets, and newly released models. Despite leaderboard records mixing evaluation protocols (zero-shot, fine-tuning, prompting), the aggregated collaborative information proves useful: integrating MODELLENS's top-K outputs with modern routing methods yields gains of up to 81% on QA benchmarks, and case studies on two recently released benchmarks confirm cross-modal transfer to text and vision-language tasks.

Our contributions are threefold: 1) We first formalize the problem of model recommendation in the wild and curate a large-scale benchmark of model–dataset–metric interactions, covering tens of thousands of models and diverse datasets across multiple domains and modalities. 2) We propose a unified, metric-aware ranking framework that leverages heterogeneous metadata to predict model-dataset compatibility, generalizing to unseen models and datasets without any direct evaluation or finetuning. 3)We show that the framework not only attains strong ranking performance, but also yields high-quality candidate sets directly compatible with downstream routing and ensemble systems, enabling scalable model selection in dynamic, large-scale ecosystems.

2 Related Works

Transferability Estimation. Transferability estimation predicts how well a pretrained model will transfer to a target task without full fine-tuning. Training-free methods estimate transferability from a single forward pass on the target dataset using information-theoretic or likelihood-based statistics [10, 17, 18, 19, 20, 21, 22, 23, 24], while learning-based approaches model interactions between feature representations and target data [11, 12]. Despite their effectiveness, TE methods assume a controlled pretrain-to-finetune pipeline and require per-model execution on the target dataset, which becomes infeasible as model hubs scale to tens of thousands of candidates [2, 3]. MODELLENS instead studies model recommendation *in the wild*: models are already fully specified systems, and rankings are predicted directly from large-scale leaderboard interactions and metadata, with forward-pass features supported as optional augmentations. A full taxonomy of TE (Section A.2.2).

Automated Model Search. Automated machine learning (AutoML) aims to automate model selection and hyperparameter tuning for a target task. Classical approaches frame this as a search or meta-learning problem over a fixed pool of pipelines or architectures [4, 8], with recent work extending this paradigm to pretrained model selection [7, 9]. While effective in curated settings, these methods assume a predefined and relatively small candidate pool, which fundamentally limits their applicability to the open and continuously evolving model ecosystems we target in this work.

Model Routing. Model routing addresses an orthogonal problem: given a *fixed* pool of candidates and an incoming query, decide which model should serve it [13, 14, 15, 25, 26, 27]. These methods take the candidate pool as given, leaving open the upstream question of how the pool itself should be constructed from a large, heterogeneous model space [28]. Our work is complementary: MODELLENS produces high-quality, task-specific candidate pools at the dataset level, which can be directly consumed by any instance-level router.

3 ModelLens

MODELLENS is a ranking framework that predicts the relative performance of candidate models on a target dataset using heterogeneous metadata, without running any candidate on the target dataset. Its design follows a single principle: combine structured inductive bias with flexible interaction modeling. Three components instantiate this principle. First, MODELLENS builds *multi-view representations* for models and datasets from learned IDs, tokenized names, and frozen text-description embeddings, supporting both memorization and generalization. Second, it conditions on the *evaluation context* (task and metric) and on structural model attributes (scale and architecture family). Third, it computes compatibility via an additive decomposition into a *structural prior* for predictable regularities such as neural scaling, and a *residual interaction* term for fine-grained model–dataset compatibility. An *ID dropout* mechanism applied during training enables zero-shot ranking on entirely new models or datasets. We first formalize the problem setting, then describe each component in turn.

3.1 Problem Definition

Let $\mathcal{M} = \{m_1, \ldots, m_N\}$ denote a large and evolving pool of available models, and $\mathcal{D} = \{d_1, \ldots, d_T\}$ a collection of datasets. Each pair (m_i, d_j) is associated with a performance score $y_{ij} \in \mathbb{R}$ under a task-specific evaluation metric, forming a performance matrix $\mathbf{Y} \in \mathbb{R}^{N \times T}$ whose observed entries are

$$\mathcal{O} = \{(m_i, d_j, y_{ij}) \mid (i, j) \in \Omega\}, \tag{1}$$

In practice, $\mathbf{Y}$ is sparse and heterogeneous: few pairs are evaluated, metrics differ across datasets so absolute scores are not directly comparable. Given a target dataset d^* with limited or no observed evaluations, the goal of *model recommendation in the wild* is to learn a scoring function

$$f : \mathcal{M} \times \mathcal{D} \times \mathcal{T} \times \mathcal{U} \to \mathbb{R}, \tag{2}$$

where $\mathcal{T}$ and $\mathcal{U}$ are the spaces of task types and evaluation metrics (necessary since the same pair can rank differently under different metrics, e.g., accuracy vs. F1). For a target dataset d^* evaluated under metric μ^* and task t^*, the framework produces:

$$m^* = \arg\max_{m \in \mathcal{M}} f(m, d^*, t^*, \mu^*), \qquad \mathcal{M}_K = \mathrm{TopK}_{m \in \mathcal{M}}\, f(m, d^*, t^*, \mu^*). \tag{3}$$

Crucially, f takes only model and dataset descriptors together with the evaluation context (t, μ) as input, and does *not* consume any feature, gradient, or forward-pass signal extracted from d^*. Since metrics are incompatible across datasets, we supervise f via the *relative ordering* of models within each evaluation group $g = (d, t, \mu) \in \mathcal{G}$, where $\mathcal{G}$ denotes the set of all (dataset, task, metric) groups observed in training rather than their absolute values, and the central challenge is to generalize this ranking to unseen models and datasets under sparse, heterogeneous observations $\mathcal{O}$.

3.2 Feature Representation

Model representation. Each model m is encoded as the concatenation of three complementary parts:

$$\mathbf{h}_m = [\, \mathbf{e}_m^{\mathrm{id}} \parallel \mathbf{e}_m^{\mathrm{name}} \parallel \mathbf{e}_m^{\mathrm{desc}} \,], \tag{4}$$

where $\mathbf{e}_m^{\mathrm{id}} \in \mathbb{R}^{d_{\mathrm{id}}}$ is a learned ID embedding that captures model-specific behaviors observed during training, $\mathbf{e}_m^{\mathrm{name}} \in \mathbb{R}^{d_{\mathrm{tok}}}$ is a compositional name embedding obtained by tokenizing the model name and aggregating token embeddings, and $\mathbf{e}_m^{\mathrm{desc}} \in \mathbb{R}^{d_{\mathrm{desc}}}$ is a frozen semantic embedding of the model's textual description using a pretrained text encoder.

Dataset representation. Each dataset d is represented as:

$$\mathbf{h}_d = [\, \mathbf{e}_d^{\mathrm{id}} \parallel \mathbf{e}_d^{\mathrm{desc}} \,], \tag{5}$$

where $\mathbf{e}_d^{\mathrm{id}} \in \mathbb{R}^{d_{\text{ds-id}}}$ is a learned dataset ID embedding, and $\mathbf{e}_d^{\mathrm{desc}} \in \mathbb{R}^{d_{\text{ds-desc}}}$ is a frozen semantic embedding of the dataset description from the same text encoder.

Evaluation context and structural attributes. Beyond the model–dataset pair, performance also depends on *how* a model is evaluated and on *what kind* of model it is. We encode the task type t and metric μ as learned embeddings $\mathbf{e}_t \in \mathbb{R}^{d_{\mathrm{task}}}$ and $\mathbf{e}_\mu \in \mathbb{R}^{d_{\mathrm{metric}}}$, allowing the score to adapt to different evaluation protocols. We further encode two structural attributes of each model: its scale, discretized into size buckets and mapped to an embedding $\mathbf{e}_m^{\mathrm{size}} \in \mathbb{R}^{d_{\mathrm{size}}}$, capturing the non-linear and task-dependent effects of neural scaling; and its architecture family, represented by an embedding $\mathbf{e}_m^{\mathrm{fam}} \in \mathbb{R}^{d_{\mathrm{fam}}}$, encoding shared inductive biases among models derived from the same architecture.

3.3 Scoring Function: Residual + Prior Decomposition

The compatibility score is decomposed additively into a *structural prior* that depends only on model attributes and a *residual interaction* that depends on the full evaluation context. This separates two complementary sources of signal: predictable performance trends from model structure, and context-dependent affinity that cannot be explained by structure alone.

Structural Prior. The structural prior $s_{\mathrm{prior}}(m)$ models the intrinsic competence of a model based solely on its structural attributes, independent of any specific dataset or task. It is parameterized as a shared function over model size and architecture family:

$$s_{\mathrm{prior}}(m) = \mathrm{MLP}_{\mathrm{prior}}\big([\, \mathbf{e}_m^{\mathrm{size}} \parallel \mathbf{e}_m^{\mathrm{fam}} \,]\big) \in \mathbb{R}. \tag{6}$$

This component explicitly models structural performance trends, such as neural scaling effects [29], as a learnable function of model structure. Unlike per-model bias terms in collaborative filtering, s_{prior} is a shared parametric function over the (size, family) space, enabling generalization to unseen models by interpolating over this space. By capturing predictable global patterns, the prior reduces the burden on the interaction model so that the residual can focus on fine-grained deviations.

Residual Interaction. The residual term $s_{\text{residual}}(m, d, t, \mu)$ models the deviation from the structural prior conditioned on the full evaluation context, capturing dataset-specific specialization and metric-dependent behavior. We concatenate all features into a joint input,

$$\mathbf{x} = [\, \mathbf{h}_m \parallel \mathbf{h}_d \parallel \mathbf{e}_m^{\text{size}} \parallel \mathbf{e}_m^{\text{fam}} \parallel \mathbf{e}_t \parallel \mathbf{e}_\mu \,], \tag{7}$$

which is passed through a multi-layer perceptron backbone to produce a hidden representation $\mathbf{h}$, followed by two linear heads:

$$\mathbf{h} = \text{MLP}_{\text{backbone}}(\mathbf{x}), \qquad s_{\text{residual}} = \mathbf{w}_{\text{pair}}^\top \mathbf{h}. \tag{8}$$

The size and family embeddings are shared across both the prior and residual pathways: while the prior captures their *marginal* effects, the residual captures *interaction* effects, such as how the benefit of model scale varies across datasets or metrics. In addition to the pairwise ranking score, the backbone also produces an auxiliary pointwise prediction:

$$\hat{z} = \mathbf{w}_{\text{point}}^\top \mathbf{h}, \tag{9}$$

which estimates the standardized performance of a model on a dataset. This auxiliary objective encourages the shared representation to be informative for both ranking and regression.

Score Composition. The final compatibility score combines the two components and rescales them by a learnable temperature:

$$\tilde{s}(m, d, t, \mu) = \frac{s_{\text{residual}}(m, d, t, \mu) + s_{\text{prior}}(m)}{\max(\tau,\, \epsilon)}. \tag{10}$$

The learnable temperature τ controls the sharpness of the result ranking distribution and ϵ is a constant to ensure numerical stability. The additive form also yields an interpretable decomposition: a model can be selected because of its general competence (s_{prior}) and its task-specific affinity (s_{residual}).

3.4 Generalization via ID Dropout

Learned ID embeddings $\mathbf{e}_m^{\text{id}}, \mathbf{e}_d^{\text{id}}$ are powerful for memorization but useless for unseen entities at training time. To prevent the model from over-relying on them, during training we independently replace each ID embedding with a shared learnable [UNK] vector with probabilities p_m and p_d:

$$\tilde{\mathbf{e}}_m^{\text{id}} = \begin{cases} \mathbf{e}_{[\text{UNK}]}^{\text{model}} & \text{with probability } p_m, \\ \mathbf{e}_m^{\text{id}} & \text{otherwise,} \end{cases} \qquad \tilde{\mathbf{e}}_d^{\text{id}} = \begin{cases} \mathbf{e}_{[\text{UNK}]}^{\text{dataset}} & \text{with probability } p_d, \\ \mathbf{e}_d^{\text{id}} & \text{otherwise.} \end{cases} \tag{11}$$

This trains a single set of parameters under two regimes simultaneously: a memorization regime when IDs are visible, and a semantic regime where the model must rely on names, descriptions, and structural attributes. At inference, unseen entities map to [UNK] and are handled without any architectural change.

3.5 Multi-Objective Learning

We supervise MODELLENS with three complementary objectives: pairwise comparisons capture local preferences, listwise likelihoods capture global ranking structure, and a pointwise regression captures absolute performance signals.

Pairwise ranking loss. Within each evaluation group, we sample pairs (m^+, m^-) where m^+ outperforms m^- and apply the BPR objective [30]:

$$\mathcal{L}_{\text{pair}} = \mathbb{E}\big[-\log \sigma\big(\tilde{s}(m^+, d) - \tilde{s}(m^-, d)\big) \big]. \tag{12}$$

Listwise ranking loss. For each evaluation group with M candidate models indexed in decreasing order of ground-truth performance, we adopt the Plackett–Luce likelihood [31, 32]:

$$\mathcal{L}_{\text{list}} = \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} \frac{1}{M_g} \sum_{i=1}^{M_g} \left[\log \sum_{j=i}^{M_g} \exp(\tilde{s}(m_j, d_g)) - \tilde{s}(m_i, d_g) \right]. \tag{13}$$

Pointwise regression loss. The auxiliary regression head is supervised against the standardized score $z(m, d)$, computed by z-scoring raw performance within each evaluation group; this within-group normalization is what makes scores comparable across heterogeneous metrics:

$$\mathcal{L}_{\text{point}} = \mathbb{E}\Big[\big(\hat{z}(m, d) - z(m, d)\big)^2 \Big]. \tag{14}$$

Final objective. The overall training objective is a weighted combination of the three losses:

$$\mathcal{L} = \lambda_{\text{list}} \mathcal{L}_{\text{list}} + \lambda_{\text{pair}} \mathcal{L}_{\text{pair}} + \lambda_{\text{point}} \mathcal{L}_{\text{point}}. \tag{15}$$

The pairwise and listwise losses operate on $\tilde{s}$ and jointly train both the prior and residual pathways, while the pointwise loss grounds the shared backbone in absolute performance magnitudes.

4 Experiments

In the experiments, we aim to answer the following questions:

Q1: Can our method accurately model model–dataset interactions, both in terms of recovering missing entries and generalizing to unseen datasets and models?

Q2: How does our method perform under standard transferability-based model selection settings?

Q3: Can dataset-level model recommendation improve instance-level routing?

4.1 Dataset Construction

We construct a large-scale dataset for *Model Recommendation in the Wild*, where the goal is to rank candidate models for a given dataset without direct evaluation. Unlike prior work focused on small or single-domain settings, our dataset captures heterogeneous model–dataset interactions across diverse tasks and modalities. We aggregate records from three public sources: HuggingFace Model Hub [3], Open LLM Leaderboard [2], and PapersWithCode [1], with HuggingFace records extracted via a three-tier pipeline prioritizing structured YAML, model-card metadata, and LLM-parsed README tables in decreasing reliability [33]. After deduplication, the dataset contains 1.62M records over 47K models and 9.6K datasets, spanning 2,551 tasks, and 348 architecture families across multiple domains. To evaluate generalization, we support two complementary settings: *performance completion*, where masked entries from observed datasets are predicted, and *cold-start generalization*, where 609 datasets and 375 models (temporally partitioned due to public released timestamps) are held out entirely from training. Dataset and model splits are further stratified across task type and modality to reduce domain skew. Full details are in Section A.3.

4.2 Model Recommendation in the Wild

Baselines and Evaluation Metrics. We compare against model selection methods from two paradigms, depending on whether they require running candidates on the target dataset. *Feature-based transferability methods* compute per-model scores from a forward pass on the target dataset, including training-free metrics (`H-Score` [17], `NCE` [18], `LEEP` [19], `NLEEP` [20], `LogME` [10], `PACTran` [21], `OTCE` [22], `LFC` [23], `GBC` [24]) and learning-based meta-rankers (`Model-Spider` [11], `Know2Vec` [12]). *Feature-free methods* rely on metadata or learned interactions: `Task2Vec` [34], `ZAP` [7], and two practitioner-heuristic strawmen, `Model Size` (parameter count) and `Model Popularity` (HuggingFace downloads). Details in Section A.5. We evaluate ranking quality using Kendall's weighted τ_w [35] as the primary metric, which emphasizes top-rank correctness, and further report *Hit@*K, *NDCG@*K, and *Rec@*K, averaged per dataset across the test set.

4.2.1 Performance Completion and Cold-start Generalization (Q1)

Setup. We evaluate our method under two complementary settings: (1)*Performance Completion.* From a partially observed performance matrix over 2,967 datasets, we randomly mask a subset of observed entries and train the model to predict their values, then derive a full ranking over candidate models from the predicted scores. This setting evaluates whether the model can recover global interaction structure from incomplete observations. (2)*Cold-start Generalization.* We further evaluate two extrapolation scenarios: *Unseen datasets* and *Unseen models*, each requiring the model

Table 1: Model ranking performance under new model and new dataset evaluation settings. Best results are in bold, and second-best results are underlined.

Setting	Method	τ_w	NDCG@1	Hit@1	Rec@1	NDCG@10	Hit@10	Rec@10	NDCG@30	Hit@30	Rec@30
Performance Completion (2967 datasets)	**ModelLens**	**0.868**	**0.954**	**0.153**	**0.153**	**0.967**	**0.521**	**0.452**	**0.974**	**0.840**	**0.764**
	ZAP	0.763	0.903	0.115	0.115	0.922	0.517	0.369	0.937	0.807	0.642
	Task2Vec	0.417	0.847	0.132	0.132	0.869	0.361	0.315	0.884	0.646	0.500
	Model Size	-0.021	0.625	0.032	0.032	0.716	0.129	0.167	0.775	0.415	0.399
	Model Popularity	-0.035	0.716	0.016	0.016	0.704	0.078	0.071	0.724	0.213	0.212
New Datasets (609 datasets)	**ModelLens**	**0.745**	**0.910**	**0.266**	**0.266**	**0.951**	**0.456**	**0.303**	**0.962**	**0.666**	**0.631**
	ZAP	0.253	0.852	0.060	0.060	0.861	0.189	0.270	0.870	0.543	0.482
	Task2Vec	0.227	0.691	0.008	0.008	0.778	0.221	0.129	0.817	0.365	0.381
	Model Size	0.059	0.621	0.036	0.036	0.721	0.117	0.190	0.780	0.430	0.421
	Model Popularity	-0.104	0.744	0.017	0.017	0.707	0.072	0.058	0.720	0.183	0.205
New Models (375 models)	**ModelLens**	**0.402**	**0.929**	**0.009**	**0.009**	**0.923**	**0.137**	**0.210**	**0.932**	**0.412**	**0.480**
	ZAP	0.307	0.884	0.004	0.004	0.913	0.072	0.165	0.920	0.299	0.469
	Task2Vec	0.078	0.844	0.000	0.000	0.870	0.109	0.139	0.879	0.310	0.250
	Model Size	0.055	0.674	0.005	0.005	0.807	0.086	0.097	0.853	0.347	0.464
	Model Popularity	-0.296	0.861	0.004	0.004	0.858	0.088	0.109	0.839	0.262	0.251

Table 2: Seen datasets model selection performance measured by Kendall's weighted τ_w.

Method	Aircraft	Cars	DTD	Pets	Flowers102	Food101	Country211	EuroSAT	Avg.
Feature-based Transferability Methods									
H-Score [17]	0.328	0.616	0.395	0.610	-0.200	0.200	-0.629	-0.067	0.157
NCE [18]	0.501	0.771	0.403	0.696	-0.200	-0.378	**0.511**	-0.200	0.263
LEEP [19]	0.244	0.704	-0.111	0.680	-0.111	-0.022	0.074	-0.244	0.152
NLEEP [20]	-0.725	0.622	0.074	0.787	0.244	-0.378	-0.422	-0.156	0.005
LogME [10]	0.540	0.677	0.429	0.628	-0.511	0.067	0.422	-0.422	0.229
PACTran [21]	0.031	0.665	-0.236	0.616	0.022	-0.067	-0.270	0.067	0.104
OTCE [22]	-0.241	-0.157	-0.165	0.402	-0.111	-0.289	-0.405	0.333	-0.079
LFC [23]	0.279	0.243	-0.722	0.215	0.244	0.467	0.405	0.478	0.201
GBC [24]	-0.744	-0.265	-0.102	0.163	0.289	-0.022	0.384	-0.200	-0.062
Model-Spider [11]	0.467	0.644	0.556	0.689	-0.556	0.067	0.244	0.289	0.3
Know2Vec [12]	0.111	0.283	0.200	0.200	0.067	-0.156	0.289	0.244	0.155
Feature-free Methods									
Task2Vec [34]	0.272	0.404	-0.279	0.426	-0.263	-0.511	-0.422	0.460	0.011
ZAP [7]	0.244	0.188	0.244	0.246	0.067	0.378	0.315	0.156	0.229
GPT-5	0.142	0.249	0.002	0.727	-0.347	**0.602**	0.355	-0.125	0.201
Ours									
ModelLens (Feature Free)	0.378	0.556	0.289	0.511	0.156	0.422	0.378	0.263	0.369
ModelLens (Feature Aug.)	**0.556**	**0.778**	**0.689**	**0.802**	**0.422**	0.556	0.467	**0.600**	**0.609**

to generalize beyond observed interactions. At this scale, feature-based transferability estimation methods are computationally infeasible since they require a forward pass per candidate on each target dataset; we therefore restrict comparison to feature-free baselines.

Results. Table 1 shows that MODELLENS consistently outperforms all baselines across both performance completion and cold-start settings. The advantage is most pronounced on unseen datasets, where baselines degrade sharply while MODELLENS remains strong, indicating that the learned representations transfer beyond observed pairs to entirely new datasets and models. A per-task-family breakdown, including where MODELLENS falls short, is given in Appendix A.8.

4.2.2 Transferability-based Model Selection (Q2)

Setups. For completeness, we also evaluate MODELLENS under the standard transferability-based model selection protocol [10], on 8 widely-used vision benchmarks: Aircraft [36], Cars [37], DTD [38], Pets [39], Flowers102 [40], Food101 [41], Country211 [42], and EuroSAT [43]. The first four are in-distribution for learning-based meta rankers, while the latter four are unseen, allowing us to probe both regimes. We use GPT-5 [44] for direct recommendation, serving as another feature-free baseline. Ranking quality is measured by Kendall's weighted τ, with MRR results in Table 15.

Feature augmentation with transferability signals and Results. We further investigate whether forward-pass features from candidate models provide complementary information. We extract intermediate representations from candidate models on the target dataset (as in feature-based baselines)

Table 4: Routing performance w.r.t Exact Match. Each method is evaluated with its original pool and with new model pool (*Recommended Pool*).

Method	NQ	PopQA	HotpotQA	Musique	Bamboogle	Avg.
KNNRouter [25]	0.262	0.222	0.224	0.066	0.360	0.227
+ Recommended Pool	**0.487 (↑85.9%)**	**0.537 (↑141.9%)**	**0.330 (↑47.3%)**	**0.101 (↑53.0%)**	**0.600 (↑66.7%)**	**0.411 (↑81.1%)**
MLPRouter [26]	0.252	0.222	0.198	0.072	0.360	0.221
+ Recommended Pool	**0.475 (↑88.5%)**	**0.490 (↑120.7%)**	**0.251 (↑26.8%)**	**0.096 (↑33.3%)**	**0.520 (↑44.4%)**	**0.366 (↑65.6%)**
RouterDC [13]	0.278	0.282	0.244	0.080	0.504	0.278
+ Recommended Pool	**0.325 (↑16.9%)**	**0.389 (↑37.9%)**	**0.350 (↑43.4%)**	**0.115 (↑43.8%)**	**0.512 (↑1.6%)**	**0.338 (↑21.6%)**
GraphRouter [27]	0.276	0.280	0.234	0.076	0.448	0.263
+ Recommended Pool	**0.405 (↑46.7%)**	**0.600 (↑114.3%)**	**0.264 (↑12.8%)**	**0.132 (↑73.6%)**	**0.584 (↑30.4%)**	**0.397 (↑51.0%)**
Router-R1-Qwen [15]	0.388	0.384	0.352	0.138	0.512	0.355
+ Recommended Pool	**0.524 (↑35.1%)**	**0.501 (↑30.5%)**	**0.538 (↑52.8%)**	**0.224 (↑62.3%)**	**0.624 (↑21.9%)**	**0.482 (↑35.8%)**

and concatenate them with our existing model representations. To prevent leakage, only auxiliary models that are disjoint from the evaluation pool contribute forward-pass features at training time, while features from evaluated candidates are used exclusively at inference. Table 2 reports per-dataset and average τ_w on the 8 benchmarks. Our method attains the best average performance among all baselines without any forward pass on the target dataset. Feature-based transferability estimators are unstable across datasets rather than uniformly weaker (e.g., NLEEP ranges from -0.725 on Aircraft to 0.787 on Pets). Adding transferability features (Feature Aug.) yields complementary gains: the average τ_w rises to 0.609, the best on six of eight datasets and on average.

4.3 Routing with Recommended Model Pools

Setups. We evaluate instance-level model routing on five QA benchmarks: NQ [54], PopQA [55], HotpotQA [56], Musique [57], and Bamboogle [58], following the setup of Router-R1 [15]. Unlike prior work that improves routing algorithms under a fixed model pool, we study the impact of *pool quality* on routing performance. We evaluate multiple routing methods, including KNNRouter [25], MLPRouter [26], RouterDC [13], GraphRouter [27], and Router-R1 [15], evaluated with both the original pool and recommended pool.

Table 3: Model pool replacement for NQ under comparable inference scale (recommended pools for other datasets in Section A.6).

Original	Scale	→	Scale	Selected
LLaMA-3.1-70B [45]	≈ 70B	→	≈ 70B	LLaMA-3.3-70B [46]
Mixtral-8x22B [47]	≈ 39B	→	≈ 20B	GPT-OSS-20B [48]
Gemma-2-27B [49]	≈ 27B	→	≈ 17B	Llama-4-Maverick [50]
LLaMA-3.1-8B [45]	≈ 8B	→	≈ 8B	Nemotron-H-8B-R [51]
Qwen2.5-7B [52]	≈ 7B	→	≈ 7B	Qwen2.5-7B [52]
Mistral-7B [53]	≈ 7B	→	≈ 4B	Gemma-3n-E4B [49]

Model Pool Construction. We start from the six-model pool released with Router-R1 [15] (the *original pool*). For each test dataset (held out from training), MODELLENS predicts model rankings from the dataset's textual description and evaluation metric alone, without access to any ground-truth performance. We then replace each model in the original pool with a top-ranked alternative of comparable scale that is available via the Together AI API[1] and matched on inference cost (parameter count for dense models, active parameters for MoE), ensuring both competitive ranking quality and deployability. Table 3 illustrates the resulting NQ pool; comparisons against recency-based and ZAP-selected pools are in Section A.7. The procedure is orthogonal to the underlying router and can be applied to any existing method.

Results. Table 4 shows routing performance under different model pools. Replacing the original pool with our recommended pool consistently improves all five routers across all five datasets, indicating that pool quality is orthogonal to and complementary with routing algorithm design.

4.4 Ablation Study

Loss ablation and Results. We ablate the three training objectives: listwise (L), pairwise (P), and pointwise regression (Pt). The full model (L+P+Pt) achieves the best τ_w of 0.745. Removing the listwise loss causes the largest degradation ($\rightarrow$ 0.632), confirming that global ranking structure is the dominant supervision signal; removing pairwise yields a moderate drop ($\rightarrow$ 0.703), and removing pointwise the smallest ($\rightarrow$ 0.728), indicating that it primarily serves as calibration. Single-loss

[1] `https://www.together.ai/`

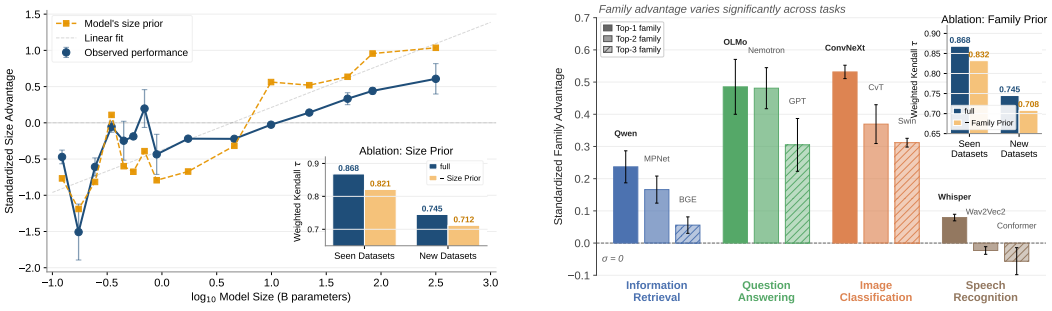

Figure 2: Learned size and family priors from model–dataset interactions. (Left) Model performance exhibits a monotonic trend with respect to model size, with higher variance in the small-model regime. (Right) Family-level advantages vary across task domains, showing strong effects in some tasks (e.g., QA, IR, vision) but weaker structure in others (e.g., speech). Ablations confirm that both priors contribute to model recommendation performance.

variants underperform all multi-loss combinations, showing the three signals are complementary. Full results are in Table 16. Further analyses of model-side and dataset-side feature contributions and unseen-family generalization are provided in Sections A.9 and A.10.

Learned Size and Family Priors. We analyze whether MODELLENS captures structured patterns from interactions, focusing on size and family effects. To enable comparison across heterogeneous tasks and metrics, we standardize performance via z-scores within each evaluation group and report group-level averages as size or family advantage (Section A.11). *Size prior.* Figure 2(left) shows a monotonic relationship between model size and predicted performance, aligning with empirical scaling trends. The effect is less stable for small models ($<$ 1B), where the regime is dominated by specialized models (e.g., vision models on vision tasks). *Family prior.* Figure 2(right) shows strong family-level effects in information retrieval, question answering, and image classification, where certain families consistently dominate; the effect is weaker in speech, where families perform more uniformly. *Impact on recommendation.* The two priors are complementary: size provides a global trend that becomes reliable at scale, while family captures task-dependent structure that varies by domain. Removing either degrades performance (Figure 2, ablation insets), confirming that both global and task-specific structure are necessary for accurate model recommendation.

4.5 Case Study: Cross-Domain Model Recommendation

We further conduct two case studies on recently released benchmarks not in our training corpus, each probing a different aspect of MODELLENS: *NGQA* [59] tests whether MODELLENS produces useful task-specific recommendations that beat practical defaults, and *RSVLM-QA* [60] tests whether its ranking is accurate beyond top-1 and generalizes to unseen candidates. The two benchmarks span text and vision-language modalities, providing a robust platform to examine cross-domain transfer.

Case 1: NGQA (Text-based Reasoning). NGQA spans three tasks over nutritional knowledge: binary classification, multi-label classification, and free-form text generation. We construct a controlled pool of models under 20B parameters (matching the implied scale of the default `GPT-4o-mini`). As shown in Figure 3(left), the optimal model varies across tasks, and the MODELLENS recommended top-1 consistently outperforms the default `GPT-4o-mini` in all settings, indicating that model suitability is task-dependent even within a single dataset, and MODELLENS captures this variation rather than committing to a single fixed choice.

Case 2: RSVLM-QA (Vision-Language Understanding). We rank eight comparable-scale (7B–8B) vision-language models on the RSVLM-QA captioning subset, of which five appear in the original benchmark and three are surfaced by MODELLENS but not evaluated there. Figure 3(right) plots MODELLENS scores against METEOR. Our method recovers the exact empirical ranking (τ=ρ=1.00), with `Ovis2` correctly identified as the best (METEOR = 31.65). Crucially, the three discovered models that are absent from the original benchmark fall precisely on the empirical regression trend, indicating that MODELLENS not only orders enumerated candidates correctly but also generalizes to identify additional competitive models without any direct evaluation.

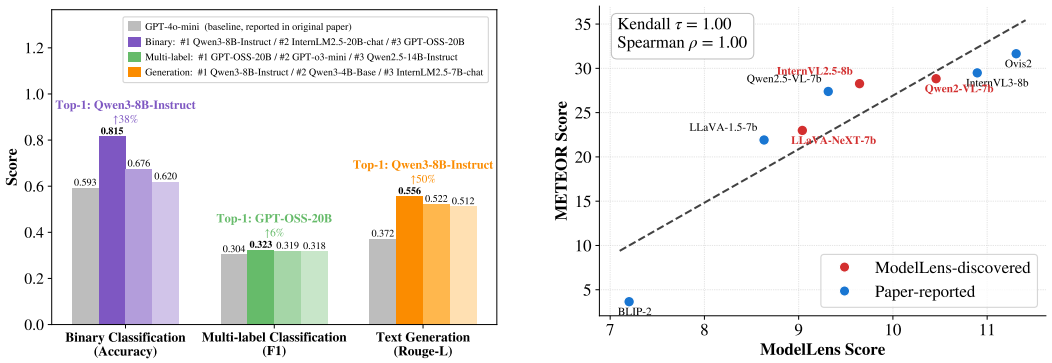

Figure 3: Case studies on unseen datasets across domains. (Left) On NGQA, different tasks favor different models, and the recommendations consistently outperform the default baseline. (Right) On RSVLM-QA, predicted scores perfectly match empirical performance, recovering the full ranking and generalizing to additional competitive models not included in the original benchmark.

5 Conclusion

We studied *model recommendation in the wild*, the problem of identifying suitable models for a target task at the scale of today's open-source ecosystem. We introduced MODELLENS, a metric-aware ranking framework that learns directly from large-scale model–dataset–metric interactions and generalizes zero-shot to unseen models and datasets. On a benchmark of 1.62M evaluation records spanning 47K models and 9.6K datasets, MODELLENS surpasses both metadata-only and forward-pass-based transferability baselines without ever running a candidate on the target task, and its recommended Top-K pools translate into 21–81% average gains across 5 representative routing methods. Beyond ranking, MODELLENS yields a learned capability profile for each of the 47K models in our corpus, supporting downstream analyses of model strengths, blind spots, and family-level trends. More broadly, our results suggest that the heterogeneous records accumulating in public leaderboards form a learnable capability atlas in their own right, a foundation layer for routing, ensembling, and model search as the open-model ecosystem expands.

Acknowledgments

We appreciate the reviewers for their constructive comments and suggestions. Rui Cai and Zhe Zhao were supported by NSF grant OAC 2531126 and a Google Research Award. Wenjie Jacky Mo, Xiaofei Wen and Muhao Chen were supported by a Noyce Initiative Grant, the NSF of the United States under Grants ITE 2333736 and OAC 2531126, the NIH 1R01CA323939-01, and equipment gifts from NVIDIA and DELL.

References

[1] Marcin Kardas, Piotr Czapla, Pontus Stenetorp, Sebastian Ruder, Sebastian Riedel, Ross Taylor, and Robert Stojnic. Axcell: Automatic extraction of results from machine learning papers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8580–8594, 2020.

[2] Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open llm leaderboard v2. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`, 2024.

[3] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[4] Xin He, Kaiyong Zhao, and Xiaowen Chu. Automl: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.

[5] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[6] Eliahu Horwitz, Nitzan Kurer, Jonathan Kahana, Liel Amar, and Yedid Hoshen. We should chart an atlas of all the world's models. *arXiv preprint arXiv:2503.10633*, 2025.

[7] Ekrem Öztürk, Fabio Ferreira, Hadi Jomaa, Lars Schmidt-Thieme, Josif Grabocka, and Frank Hutter. Zero-shot automl with pretrained models. In *International Conference on Machine Learning*, pages 17138–17155. PMLR, 2022.

[8] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

[9] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*, 2023.

[10] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *International conference on machine learning*, pages 12133–12143. PMLR, 2021.

[11] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. Model spider: Learning to rank pre-trained models efficiently. *Advances in Neural Information Processing Systems*, 36:13692–13719, 2023.

[12] Zhuoyi Shang, Yanwei Liu, Jinxia Liu, Xiaoyan Gu, Ying Ding, and Xiangyang Ji. Know2vec: A black-box proxy for neural network retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 20346–20353, 2025.

[13] Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024.

[14] Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. *arXiv preprint arXiv:2410.02223*, 2024.

[15] Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[16] 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*, pages 9556–9567, 2024.

[17] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *2019 IEEE international conference on image processing (ICIP)*, pages 2309–2313. IEEE, 2019.

[18] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1395–1405, 2019.

[19] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International conference on machine learning*, pages 7294–7305. PMLR, 2020.

[20] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2663–2673, 2021.

[21] Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *European Conference on Computer Vision*, pages 252–268. Springer, 2022.

[22] Yang Tan, Yang Li, and Shao-Lun Huang. Otce: A transferability metric for cross-domain cross-task representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15779–15788, 2021.

[23] Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv preprint arXiv:2102.00084*, 2021.

[24] Michal Pándy, Andrea Agostinelli, Jasper Uijlings, Vittorio Ferrari, and Thomas Mensink. Transferability estimation using bhattacharyya class separability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9172–9182, 2022.

[25] Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

[26] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

[27] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*, 2024.

[28] Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in llms. *arXiv preprint arXiv:2503.10657*, 2025.

[29] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[30] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[31] Robin L Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

[32] R Duncan Luce et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

[33] Zhengyuan Dong, Victor Zhong, and Renée J Miller. Modeltables: A corpus of tables about models. *arXiv preprint arXiv:2512.16106*, 2025.

[34] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6430–6439, 2019.

[35] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1-2):81–93, 1938.

[36] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[37] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[38] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[39] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[40] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[41] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *European conference on computer vision*, pages 446–461. Springer, 2014.

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[43] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[44] OpenAI. Introducing gpt-5, August 2025.

[45] Meta AI. Llama 3.1 model card. `https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/MODEL_CARD.md`, 2024. Accessed: 2026-04-13.

[46] Meta AI. Llama 3.3 model card. `https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md`, 2024. Accessed: 2026-04-13.

[47] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts. *CoRR*, abs/2401.04088, 2024.

[48] OpenAI. Gpt-oss-20b. `https://huggingface.co/openai/gpt-oss-20b`, 2025. Accessed: 2026-05.

[49] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[50] Meta AI. Llama 4 model card. `https://www.llama.com/docs/model-cards-and-prompt-formats/llama4/`, 2025. Accessed: 2026-04-13.

[51] Aaron Blakeman, Aarti Basant, Abhinav Khattar, Adithya Renduchintala, Akhiad Bercovich, Aleksander Ficek, Alexis Bjorlin, Ali Taghibakhshi, Amala Sanjay Deshmukh, Ameya Sunil Mahabaleshwarkar, et al. Nemotron-h: A family of accurate and efficient hybrid mamba-transformer models. *arXiv preprint arXiv:2504.03624*, 2025.

[52] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report. *CoRR*, abs/2501.15383, 2025.

[53] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

[54] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019.

[55] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics, 2023.

[56] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018.

[57] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022.

[58] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, Findings of ACL, pages 5687–5711. Association for Computational Linguistics, 2023.

[59] Zheyuan Zhang, Yiyang Li, Nhi Ha Lan Le, Zehong Wang, Tianyi Ma, Vincent Galassi, Keerthiram Murugesan, Nuno Moniz, Werner Geyer, Nitesh V. Chawla, Chuxu Zhang, and Yanfang Ye. NGQA: A nutritional graph question answering benchmark for personalized health-aware nutritional reasoning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 5934–5966. Association for Computational Linguistics, 2025.

[60] Xing Zi, Jinghao Xiao, Yunxiao Shi, Xian Tao, Jun Li, Ali Braytee, and Mukesh Prasad. RSVLM-QA: A benchmark dataset for remote sensing vision language model-based question answering. In Cathal Gurrin, Klaus Schoeffmann, Min Zhang, Luca Rossetto, Stevan Rudinac, Duc-Tien Dang-Nguyen, Wen-Huang Cheng, Phoebe Chen, and Jenny Benois-Pineau, editors, *Proceedings of the 33rd ACM International Conference on Multimedia, MM 2025, Dublin, Ireland, October 27-31, 2025*, pages 12905–12911. ACM, 2025.

[61] Thomas Unterthiner, Daniel Keysers, Sylvain Gelly, Olivier Bousquet, and Ilya Tolstikhin. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2002.11448*, 2020.

[62] Charles H Martin, Tongsu Peng, and Michael W Mahoney. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4122, 2021.

[63] Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-representations as generative models: Sampling unseen neural network weights. *Advances in Neural Information Processing Systems*, 35:27906–27920, 2022.

[64] Damian Falk, Konstantin Schürholt, Konstantinos Tzevelekakis, Léo Meynent, and Damian Borth. Learning model representations using publicly available model hubs. *arXiv preprint arXiv:2510.02096*, 2025.

[65] Zhaomin Wu, Haodong Zhao, Ziyang Wang, Jizhou Guo, Qian Wang, and Bingsheng He. Llm dna: Tracing model evolution via functional representations. *arXiv preprint arXiv:2509.24496*, 2025.

[66] Koyena Pal, David Bau, and Renée J Miller. Model lakes. *arXiv preprint arXiv:2403.02327*, 2024.

[67] Xiaofei Wen, Wenxuan Zhou, Wenjie Jacky Mo, and Muhao Chen. ThinkGuard: Deliberative slow thinking leads to cautious guardrails. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13698–13713, Vienna, Austria, July 2025. Association for Computational Linguistics.

[68] Rui Cai, Bangzheng Li, Xiaofei Wen, Muhao Chen, and Zhe Zhao. Diagnosing and mitigating modality interference in multimodal large language models. *arXiv preprint arXiv:2505.19616*, 2025.

[69] Boyu Zhu, Xiaofei Wen, Wenjie Jacky Mo, Tinghui Zhu, Yanan Xie, Peng Qi, and Muhao Chen. Omniguard: Unified omni-modal guardrails with deliberate reasoning. *CoRR*, abs/2512.02306, 2025.

[70] Wenxiao Wang, Weiming Zhuang, and Lingjuan Lyu. Towards fundamentally scalable model selection: Asymptotically fast update and selection. *arXiv preprint arXiv:2406.07536*, 2024.

[71] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[72] Prateek Jain and Inderjit S Dhillon. Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626*, 2013.

[73] Si Si, Kai-Yang Chiang, Cho-Jui Hsieh, Nikhil Rao, and Inderjit S Dhillon. Goal-directed inductive matrix completion. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1165–1174, 2016.

[74] Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. *arXiv preprint arXiv:1904.12058*, 2019.

[75] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. *Advances in neural information processing systems*, 30, 2017.

[76] Lloyd S Shapley et al. A value for n-person games. 1953.

A Appendix

A.1 Appendix Overview

This appendix provides additional details, analyses, and reproducibility information for MODELLENS. We organize the supplementary material as follows.

A. Learned Embedding Space Figure 4–Figure 5

Visualizations of the interaction-trained and semantic-only model–dataset embedding spaces, highlighting how performance interactions induce more functional organization than textual similarity alone.

B. Related Work and Comparison Section A.2, Table 5

Detailed discussion of model profiling, transferability estimation, model routing, AutoML, and matrix-completion baselines, together with additional baseline results.

C. Dataset Construction Section A.3, Table 6

Additional statistics and preprocessing details for the *Model Recommendation in the Wild* benchmark, including data sources, the three-tier HuggingFace extraction pipeline, interaction normalization, metadata construction, and split design.

D. Implementation and Evaluation Details Section A.4

Architecture, embedding dimensions, optimization settings, ranking losses, batch construction, evaluation metrics, and compute resources used in our experiments.

E. Baseline Details Section A.5

Additional implementation details for feature-based transferability methods, feature-free methods, practitioner strawmen, and evaluation metrics.

F. Recommended Model Pools for Routing Section A.6, Table 8–Table 13

Recommended replacement pools for PopQA, HotpotQA, MuSiQue, and Bamboogle under comparable inference-scale constraints, together with the raw dataset descriptions used as semantic inputs.

G. Ablations, Priors, and Case Studies Section A.9, Section A.11

Additional ablation results, feature-attribution analysis, standardized advantage computation, learned size/family priors, and full case-study rankings.

H. Unseen-Family Generalization Section A.10

Evaluation under a strict family-level hold-out protocol, where entire modern LLM families are excluded from training and only appear at test time. This section analyzes unseen-family generalization and transferable model–dataset compatibility beyond family-specific memorization.

I. Limitations, Broader Impacts, Assets, and Reproducibility Section A.12

Discussion of limitations, broader impacts, data and asset licenses, and reproducibility information.

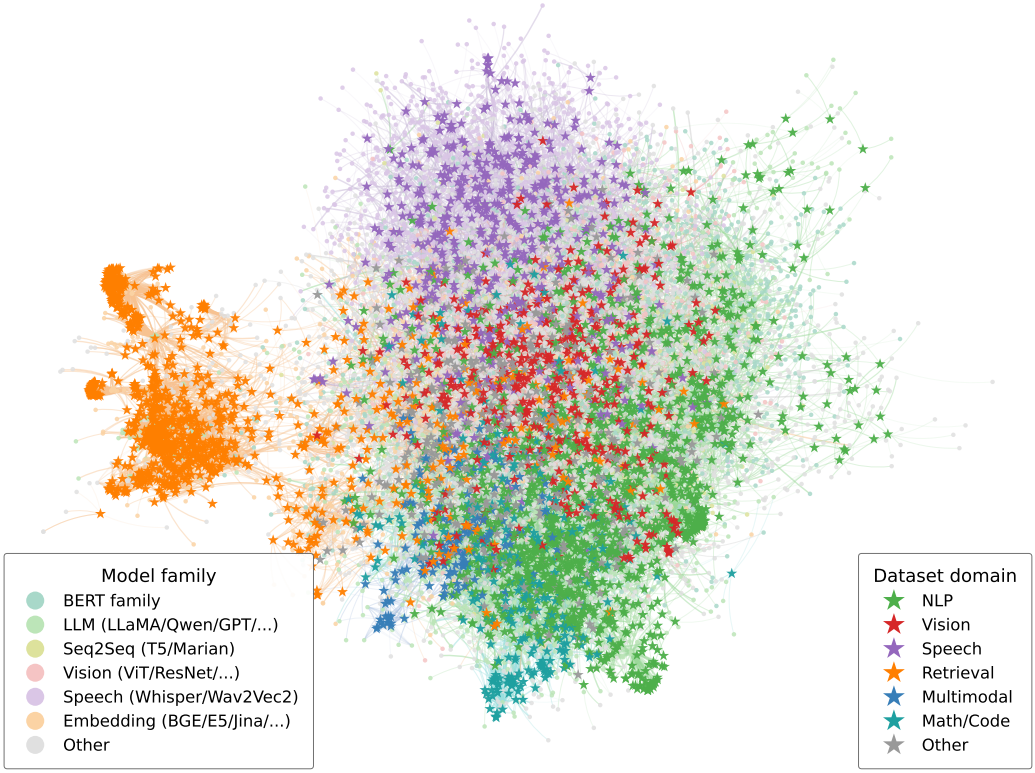

Figure 4: Visualization of the learned model–dataset embedding space trained on interaction data.Each point corresponds to either a model or a dataset, projected into a shared latent space learned from large-scale performance interactions. Compared to semantic-only representations, the learned space exhibits clear functional organization, where models and datasets cluster according to their task characteristics (e.g., NLP, vision, multimodal).This indicates that the model captures performance-aware relationships beyond surface-level similarity, enabling more meaningful grouping of models and datasets for downstream recommendation.

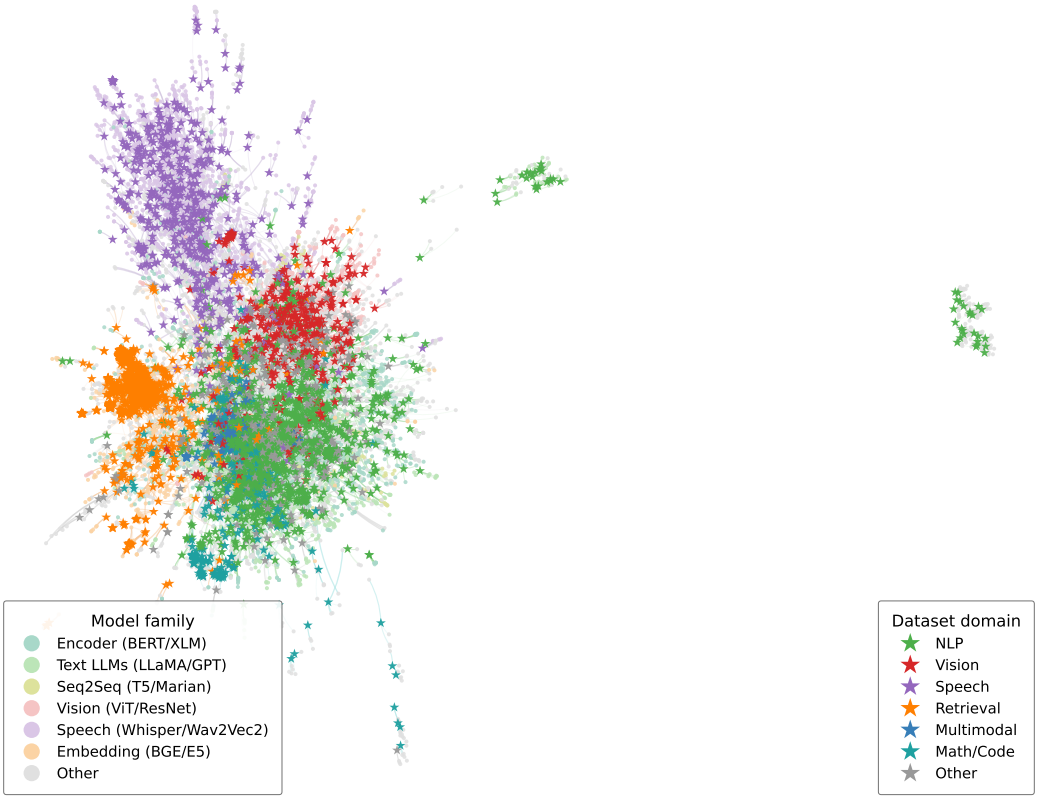

Figure 5: Visualization of the model–dataset embedding space constructed using semantic (content-based) features only. The embedding is derived from textual descriptions and metadata without leveraging performance interactions. Unlike the interaction-trained space, this representation shows limited structural organization, with different task domains intermingled and no clear clustering patterns. This highlights the limitation of relying solely on semantic similarity for model recommendation, as it fails to capture performance-relevant relationships between models and datasets.

A.2 Detailed Related Works and Baseline Comparison

A.2.1 Model Profiling

A growing body of work shifts the focus from individual models to entire model populations, treating models themselves as a data modality to support downstream applications such as model discovery and selection. We group this literature into three complementary directions.

Weight space learning. The earliest direction studies models directly through their parameters. Unterthiner et al. [61] showed that simple statistics of neural network weights suffice to predict model accuracy with high fidelity, even transferring across unseen datasets and architectures. Martin et al. [62] extended this to large pretrained models through a heavy-tailed self-regularization perspective, and Schürholt et al. [63] formalized the paradigm of *weight space learning*, proposing self-supervised representations over model zoos that capture intrinsic properties such as accuracy and hyperparameters. More recent work scales this paradigm beyond curated zoos to large, heterogeneous repositories such as HuggingFace [64]. While effective, all of these approaches require direct access to model weights and characterize models in isolation rather than their compatibility with specific tasks.

Functional representations of models. A second line of work characterizes models through their functional behavior rather than their parameters. LLM DNA [65] embeds models into a low-dimensional space based on their responses to probe inputs, enabling similarity analysis, clustering, and lineage inference. This approach avoids direct reliance on weight access and captures behavioral characteristics of models. However, such methods typically require multiple forward passes per model, which limits scalability to large candidate pools. In addition, the resulting representations reflect global model similarity rather than task-specific suitability, making them insufficient for direct model selection.

Model ecosystem analysis. A third direction studies the structural organization of model repositories. Model Atlas [6] proposes representing large model collections as a graph, where nodes correspond to models and edges capture transformations such as fine-tuning, quantization, or merging. This framework enables applications such as model forensics, lineage recovery, and meta-ML analysis over large-scale model ecosystems. Model Lakes [66] frames model attribution, versioning, search, and benchmarking over large model collections as data-management tasks, and ModelTables [33] builds a corpus of 90K tables from model cards, READMEs, and papers for structured table retrieval. While these approaches provide a structured view of model populations and support model discovery at the infrastructure level, they do not address the core decision problem of selecting models for a given task, nor do they predict task-specific performance.

Our position. Unlike prior work that represents models in isolation, we learn model–dataset *interactions* directly from large-scale leaderboard data, predicting task-aware rankings without model weights or forward passes. This applies to both open and closed models and scales to ecosystems with tens of thousands of candidates.

A.2.2 Transferability Estimation and Model Selection

Transferability estimation (TE) aims to predict how well a pre-trained model will perform on a target task without the prohibitive cost of full fine-tuning. *Training-free methods* typically estimate transferability by performing a single forward pass on the target dataset to extract feature–label statistics. Early works such as H-Score [17] and NCE [18] utilize information-theoretic measures, while LEEP [19] and NLEEP [20] extend these concepts using soft-label distributions. LogME [10] formulates transferability as a marginal likelihood problem, and subsequent research has introduced variants like PACTran [21], OTCE [22], LFC [23], and GBC [24] to address specific transfer scenarios.

Learning-based methods move beyond static metrics by leveraging interaction patterns. For instance, Model-Spider [11] employs a cross-attention meta-ranker over feature representations, and Know2Vec [12] maps per-class statistics into a shared embedding space. While effective, these approaches are fundamentally limited by their reliance on target-task inference [11, 12]. As the ecosystem expands to tens of thousands of models, running forward passes for every candidate becomes computationally infeasible [2, 3]. MODELLENS diverges from this paradigm by predicting rankings directly from leaderboard interactions and structural metadata, while retaining the flexibility to incorporate forward-pass features as optional augmentations when compute allows.

Table 5: Comparison with matrix-completion and inductive recommendation baselines under performance completion and new dataset evaluation settings. Best results are in bold, and second-best results are underlined.

Setting	Method	τ_w	NDCG@1	Hit@1	Recall@1	NDCG@10	Hit@10	Recall@10	NDCG@30	Hit@30	Recall@30
Performance Completion (2967 datasets)	**ModelLens**	**0.868**	**0.954**	**0.139**	**0.139**	**0.967**	**0.521**	**0.452**	**0.974**	0.840	**0.764**
	TwoTowerCosine	0.765	0.900	0.093	0.093	0.918	0.434	0.341	0.933	0.632	0.629
	Standardized Embedder [70]	0.167	0.661	0.036	0.036	0.717	0.138	0.096	0.755	0.237	0.258
	MF [71]	0.843	0.935	0.126	0.126	0.946	0.489	0.424	0.956	0.781	0.720
	IMC [72]	0.739	0.894	0.064	0.064	0.910	0.462	0.340	0.921	0.691	0.571
	GIMC [73]	0.503	0.776	0.078	0.078	0.823	0.319	0.214	0.864	0.652	0.498
	IGMC [74]	0.762	0.907	0.089	0.089	0.927	0.520	0.403	0.940	**0.843**	0.665
New Datasets (2764 datasets)	**ModelLens**	**0.817**	**0.908**	**0.171**	**0.171**	**0.929**	**0.471**	**0.410**	**0.943**	**0.855**	**0.672**
	TwoTowerCosine	0.798	0.900	0.097	0.097	0.918	0.418	0.335	0.935	0.631	0.638
	Standardized Embedder [70]	0.346	0.668	0.042	0.042	0.718	0.147	0.098	0.758	0.260	0.257
	MF [71]	0.726	0.879	0.047	0.047	0.904	0.324	0.268	0.923	0.586	0.609
	IMC [72]	0.792	0.900	0.058	0.058	0.916	0.448	0.345	0.926	0.712	0.580
	GIMC [73]	0.576	0.766	0.085	0.085	0.821	0.291	0.210	0.861	0.637	0.490
	IGMC [74]	0.800	0.866	0.072	0.072	0.889	0.334	0.277	0.912	0.564	0.579

A.2.3 Model Routing and Adaptive Inference

Existing routing methods typically assume the candidate pool is predefined and relatively small [28]. However, in a heterogeneous model space, the quality of this "upstream" pool significantly impacts downstream routing efficiency. MODELLENS serves as a foundational step for these systems by producing high-quality, task-specific candidate sets at the dataset level, which can then be seamlessly consumed by instance-level routers [13, 15].

A.2.4 AutoML and Surrogate Modeling

The challenge of model recommendation in the wild is closely related to Zero-shot AutoML [4]. Methods like ZAP [7] and TabPFN [8] utilize neural surrogates to predict performance across diverse tasks. Similarly, Optimus [9] leverages large language models for optimization modeling. However, these methods often struggle with the scale and modality heterogeneity inherent in modern model hubs. By incorporating structural priors inspired by neural scaling laws [29] and architectural family biases, MODELLENS explicitly reasons about model capacity [67] and multimodal robustness [68, 69], enabling more robust generalization than traditional surrogate models.

A.2.5 Matrix Completion in Recommender Systems

From a collaborative filtering perspective, model recommendation can be formulated as a sparse matrix completion problem, where the target is to populate a performance matrix $Y \in \mathbb{R}^{N \times T}$ representing N models and T datasets. Traditional Matrix Factorization (`MF`) [71] techniques excel at capturing latent interactions but are fundamentally transductive, relying on fixed identity (ID) embeddings that cannot generalize to the "cold-start" scenario of newly released models or datasets. To address this limitation, Inductive Matrix Completion (`IMC`) [72] and its Goal-directed `GIMC` [73] variants incorporate side information to enable prediction for unseen entities. Furthermore, `IGMC` [74] utilizes graph neural networks to learn inductive representations from local subgraphs, providing a powerful framework for reasoning over sparse interaction data. In the specific domain of model selection, the `TwoTowerCosine` architecture—often paired with `Task2Vec` [34] for generating task-specific embeddings—has become a standard for aligning model capabilities with task requirements in a shared latent space. Additionally, the `Standardized Embedder` [70] framework focuses on the fundamental scalability of these selections, offering asymptotically fast updates essential for maintaining myriads of models.

MODELLENS advances these inductive paradigms by introducing a dual-pathway scoring function that decomposes performance into a structural prior and a residual interaction term. While existing inductive methods like IMC rely heavily on side features, MODELLENS draws inspiration from `DropoutNet` [75] and employs a unique *ID-dropout* mechanism. This mechanism induces a *dual-mode training regime*: a "memorization mode" that utilizes learned ID embeddings for high-fidelity ranking of seen models, and a "semantic mode" that forces the model to leverage name, description, and structural attributes for zero-shot generalization to unseen entities. Unlike traditional DropoutNet applications in generic recommendation, our framework specifically integrates this mechanism with a structural prior derived from neural scaling laws [29] and architectural families By grounding the shared representation in absolute performance magnitudes via an auxiliary pointwise loss, MODELLENS ensures a more robust calibration than traditional ranking-only matrix completion.

Table 6: Summary of the *Model Recommendation in the Wild* dataset.

Category	Attribute	Value	Description	Notes
Scale	# Models	47,062	Unique pretrained models	Across multiple domains
	# Datasets	9,682	Distinct evaluation datasets	Includes vision, NLP, speech
	# Tasks	2,551	Task categories	Unified taxonomy
	# Metrics	8420	Unique evaluation metrics	Unified taxonomy
	#Interactions	1,623,284	Model–dataset–metric evaluation pairs	After deduplication
Sources	HuggingFace	1.64M (raw)	Model hub extraction	Gold/Silver/Bronze pipeline
	Open LLM Leaderboard	147K	LLM benchmarks	Dense evaluation
	PapersWithCode	10.8K	SOTA results	Sparse but diverse
Representation	Dataset Embedding	1536-d	Text encoder (Text-Embedding-3-small)	Description-based
	Model Size	21 buckets	Parameter discretization	Structural prior
	Model Family	348 categories	Architecture grouping	e.g., LLaMA, ViT
Splits	Train	1.51M	Training set	Stratified by dataset
	Validation	168K	Hyperparameter tuning	–
	Test	187K	In-distribution evaluation	–
OOD Setting	Held-out Datasets	609	Unseen datasets	No overlap with train
	OOD Interactions	746K	Evaluation pairs	Generalization test

A.3 Dataset Construction Details

We provide additional details of the data collection and preprocessing pipeline for the *Model Recommendation in the Wild* benchmark. A summary of dataset statistics, sources, representations, and splits is provided in Table 6.

Data Sources We aggregate model–dataset performance interactions from three complementary sources:

HuggingFace Model Hub. We develop a three-tier extraction pipeline to collect evaluation results from model repositories: (i) structured YAML results (`.eval_results/`), (ii) standardized `model-index` metadata in model cards, and (iii) Markdown tables in README files, which are parsed into structured tuples using an LLM-based extractor. This process yields 1.64M raw interactions across diverse pipeline tags.

Open LLM Leaderboard. We incorporate 147K evaluation interactions from 3,495 large language models across 43 benchmark datasets.

PapersWithCode. We include 10.8K interactions covering 5,443 models and 2,070 datasets from publicly reported results.

Data Processing All interactions are unified into a standard tuple (m, d, t, μ, v), representing the performance of model m on dataset d under task t and metric μ.

We apply standard preprocessing steps including deduplication across sources, normalization of dataset and metric names, and filtering of incomplete or inconsistent entries. After processing, the dataset contains 1.62M interactions.

Representation and Splits We encode dataset descriptions using pretrained text embeddings and incorporate model metadata such as parameter size and model family as structural features. Dataset-level splits are constructed via stratified sampling, and a cold-start setting is created by holding out a subset of datasets that do not appear during training. For unseen-model evaluation, held-out models are partitioned temporally according to their public release timestamps, simulating a realistic open-world setting in which newly released models must be recommended without prior interaction observations. For HuggingFace models, release timestamps are determined using the earliest public repository creation time or first available model-card timestamp.

Discussion Compared to prior benchmarks, our dataset is distinguished by its scale, heterogeneity across domains and modalities, and its grounding in real-world, noisy reporting practices from model repositories.

A.4 Implementation Details

Our recommendation framework is implemented in PyTorch and trained with a joint listwise–pairwise ranking objective. Unless otherwise specified, all experiments use the same backbone architecture, embedding configuration, and optimization settings across datasets and evaluation regimes. We describe the main implementation details below.

A.4.1 Model Architecture and Embeddings

Embedding configuration. The default full-feature configuration uses the dimensions in Table 7. Model and dataset description embeddings are pre-computed using `text-embedding-3-small` (dim 1536) and cached before training. These embeddings remain frozen during optimization. Hashed model-name tokens are represented with a trainable embedding table of dimension 512. Discrete metadata features, including model size bucket, model family, task ID, and dataset ID, are represented with lightweight learnable embeddings. The final scoring head is implemented as a two-layer MLP with hidden width 512 and dropout rate 0.02. To improve generalization to unseen models and datasets, we additionally apply learned-ID dropout ($p = 0.1$) on the model and dataset ID embeddings during training.

Component	Dimension
Model description embedding (frozen)	1536
Model name token embedding (learned)	512
Model size bucket embedding	64
Model family embedding	64
Dataset description embedding (frozen)	1536
Dataset ID embedding (learned)	256
Task ID embedding	256
MLP hidden width	512

Table 7: Embedding and hidden dimensions used in the full-feature ranker.

A.4.2 Optimization and Training

Optimizer and regularization. All models are trained using `AdamW` with learning rate 1×10^{-3} and weight decay 1×10^{-4}. We do not use a learning-rate scheduler; instead, training is controlled through early stopping with patience of 20 epochs based on validation weighted Kendall's τ. Gradients are clipped to a global ℓ_2 norm of 5.0 at every step.

Batch construction. Listwise batches contain $B_{\text{list}} = 8$ datasets, each expanded into its full candidate ranklist (typically 20–200 models). Pairwise batches contain $B_{\text{pair}} = 1024$ anchor–negative pairs. The listwise and pairwise loaders are interleaved during training so that each epoch terminates when the listwise loader is exhausted.

Random seeds. Unless otherwise specified, we report the mean and standard deviation over three random seeds. Variance reflects randomness from initialization, training order, and pairwise sampling under fixed train/validation/test splits.

A.4.3 Ranking Objectives

Target normalization. For each $(\text{task}, \text{dataset})$ group, we sort candidate models by their raw evaluation metric and apply z-score normalization within the group. This normalization mitigates heterogeneity across metrics such as accuracy, F1, EM, and MMLU score, allowing the model to learn relative ranking signals instead of absolute metric values.

Pair construction. For the pairwise objective, each anchor corresponds to a position $i \in \{0, \ldots, M-2\}$ in the ground-truth ranking. One negative is sampled uniformly from lower-ranked positions $\{i+1, \ldots, M-1\}$.

Joint ranking objective. The final ranker is trained with a joint objective:

$$\mathcal{L} = \lambda_{\text{list}}\mathcal{L}_{\text{list}} + \lambda_{\text{pair}}\mathcal{L}_{\text{pair}} + \lambda_{\text{point}}\mathcal{L}_{\text{point}},$$

where $\lambda_{\text{list}} = 0.5$, $\lambda_{\text{pair}} = 1.0$, and $\lambda_{\text{point}} = 0.1$.

Listwise objective. For a ranklist of length M with predicted scores $s_1, \ldots, s_M$ sorted by ground-truth rank, the listwise loss is:

$$\mathcal{L}_{\text{list}} = \frac{1}{M} \sum_{i=1}^{M} \left[\log \sum_{j=i}^{M} \exp(s_j/\tau) - s_i/\tau \right],$$

where the temperature is set to $\tau = 10$.

Pairwise objective. Given anchor and negative scores (s_+, s_-), we use the standard Bayesian Personalized Ranking (BPR) objective:

$$\mathcal{L}_{\text{pair}} = \mathbb{E}\left[-\log \sigma(s_+ - s_-) \right].$$

Pointwise auxiliary objective. Both branches additionally regress the predicted z-score against the normalized ground-truth value using an MSE loss. This auxiliary objective stabilizes optimization during early training.

A.4.4 Evaluation Protocol

Ranking metrics. For each $(\text{task}, \text{dataset})$ group, we score all candidate models and compare the predicted ranking against the ground-truth ranking induced by held-out leaderboard metrics. Our primary metric is weighted Kendall's τ, averaged across groups with weight $1/M$ to avoid domination by large candidate pools. We additionally report NDCG@k, Hit@k, and Recall@k for $k \in \{1, 10, 30, 50\}$. Groups with fewer than k candidates are excluded from the corresponding @k metric.

Model selection. The checkpoint reported in the main paper is selected based on the best validation weighted Kendall's τ. Checkpoints are evaluated and saved automatically at every epoch whenever validation performance improves.

A.4.5 Compute Resources

Hardware. The main model is trained using 1 A6000 GPU with PyTorch DistributedDataParallel. A full training run typically converges within approximately 6–8 hours wall-clock time due to early stopping, while evaluation on the complete test grid requires less than 5 minutes.

Scalability. Our framework operates entirely on leaderboard interactions and metadata,without requiring downstream model execution or fine-tuning during recommendation inference. Recommendation complexity scales linearly with the number of candidate models for a given dataset.

A.5 Baseline Details

Feature-based transferability methods. These methods compute a scalar score for each candidate model given a target dataset, based on feature or label statistics extracted from a forward pass. We consider both training-free and learning-based approaches. Training-free methods compute a scalar score per model from feature or label statistics obtained via a single forward pass: `H-Score` [17], `NCE` [18], `LEEP` [19], `NLEEP` [20], `LogME` [10], `PACTran` [21], `OTCE` [22], `LFC` [23], and `GBC` [24]. Learning-based methods improve ranking quality by modeling interactions between model features and target data: `Model-Spider` [11] is a cross-attention meta-learner over heterogeneous feature representations extracted on the target dataset, and `Know2Vec` [12] maps per-class feature statistics and task queries into a shared embedding space. We follow the standard evaluation protocol of Zhang et al. [11].

Feature-free methods. These methods do not require running models on the target dataset, and instead rely on dataset metadata or learned model–dataset interactions. `Task2Vec` [34] embeds datasets via the Fisher Information of a probe network and transfers rankings from the nearest training datasets. `ZAP` [7] is a neural surrogate predicting model performance from model and dataset features. Both were originally designed for small curated pools; we use the original implementations of both methods, restricting their inputs to entities present in our benchmark.

Practitioner strawmen. When no recommendation tool is available, practitioners commonly fall back on simple heuristics. We include two such baselines: `Model Size` ranks candidates purely by parameter count (reflecting the heuristic that larger models perform better), and `Model Popularity` ranks them by recent HuggingFace download counts (reflecting community-aggregated quality signals).

Evaluation metrics. We use Kendall's weighted τ_w [35] rather than standard τ because misorderings near the top of the ranking matter more than those at the tail in practice — a recommender that confuses ranks 1 and 2 is far more harmful than one confusing ranks 100 and 101. We complement it with three top-K metrics that capture different aspects of recommendation quality: *Hit@*K measures whether any truly top-ranked model appears in the top-K, *NDCG@*K measures position-weighted relevance, and *Recall@*K measures coverage of competitive models. All metrics are computed per dataset and averaged across the test set.

Table 8: Model pool replacement for PopQA [55] under comparable inference scale.

Original	Scale	→	Scale	Selected
LLaMA-3.1-70B	≈ 70B	→	≈ 70B	LLaMA-3.3-70B
Mixtral-8x22B	≈ 39B	→	≈ 20B	GPT-OSS-20B
Gemma-2-27B	≈ 27B	→	≈ 14B	Mixtral-8x7b-v0.1
LLaMA-3.1-8B	≈ 8B	→	≈ 8B	LLaMA-3-8B
Qwen2.5-7B	≈ 7B	→	≈ 4B	Gemma-3n-E4B
Mistral-7B	≈ 7B	→	≈ 3B	Trinity-Mini-Base

Table 9: Model pool replacement for HotpotQA [56] under comparable inference scale.

Original	Scale	→	Scale	Selected
LLaMA-3.1-70B	≈ 70B	→	≈ 70B	LLaMA-3.3-70B
Mixtral-8x22B	≈ 39B	→	≈ 20B	GPT-OSS-20B
Gemma-2-27B	≈ 27B	→	≈ 17B	Llama-4-Maverick
LLaMA-3.1-8B	≈ 8B	→	≈ 8B	Qwen3-8b
Qwen2.5-7B	≈ 7B	→	≈ 7B	LLaMA-3.1-8B
Mistral-7B	≈ 7B	→	≈ 3B	Trinity-Mini-Base

Table 10: Model pool replacement for Musique [57] under comparable inference scale.

Original	Scale	→	Scale	Selected
LLaMA-3.1-70B	≈ 70B	→	≈ 72B	Qwen2.5-72B
Mixtral-8x22B	≈ 39B	→	≈ 32B	Kimi-K2.5
Gemma-2-27B	≈ 27B	→	≈ 17B	Llama-4-Maverick
LLaMA-3.1-8B	≈ 8B	→	≈ 8B	Nemotron-H-8B-R
Qwen2.5-7B	≈ 7B	→	≈ 7B	Qwen2.5-7B
Mistral-7B	≈ 7B	→	≈ 4B	Gemma-3n-E4B

Table 11: Model pool replacement for Bamboogle [58] under comparable inference scale.

Original	Scale	→	Scale	Selected
LLaMA-3.1-70B	≈ 70B	→	≈ 72B	Qwen2.5-72B
Mixtral-8x22B	≈ 39B	→	≈ 32B	Kimi-K2.5
Gemma-2-27B	≈ 27B	→	≈ 22B	Qwen3-235B-A22B
LLaMA-3.1-8B	≈ 8B	→	≈ 8B	Llama-3-8B
Qwen2.5-7B	≈ 7B	→	≈ 7B	Mistral-7B
Mistral-7B	≈ 7B	→	≈ 3B	Trinity-Mini-Base

A.6 Recommended Model Pools for Routing

We present the model pools recommended by MODELLENS for several representative question answering benchmarks, including PopQA, HotpotQA, MuSiQue, and Bamboogle, in Tables 8–11. For each dataset, MODELLENS generates a replacement pool under comparable inference scale constraints, where candidate models are selected based on predicted compatibility with the target dataset semantics rather than direct evaluation on the benchmark itself. The resulting pools illustrate how MODELLENS adapts model selection to different reasoning demands, factual recall requirements, and robustness characteristics across datasets.

To improve reproducibility, we additionally provide in Table 12 the exact raw dataset descriptions used as semantic metadata inputs for pool generation. These descriptions are directly consumed by the routing framework to construct dataset representations and infer capability requirements for model recommendation, without manual feature engineering or benchmark-specific heuristics.

Table 12: Raw dataset descriptions used as semantic inputs for model recommendation.

Dataset	Task	Raw Description
HotpotQA	Question Answering	HotpotQA is a large-scale question answering dataset specifically designed to evaluate multi-hop reasoning and explainable question answering systems. Unlike single-document QA benchmarks, HotpotQA requires models to jointly reason over multiple documents to derive the correct answer, reflecting more complex and realistic information-seeking scenarios. The dataset is constructed from Wikipedia articles and contains over 110,000 question-answer pairs, each associated with multiple supporting documents. Crucially, HotpotQA provides explicit supporting fact annotations, identifying the exact sentences across different documents that are necessary for answering each question. This design enables fine-grained supervision not only on answer correctness but also on the reasoning process itself.
NQ	Question Answering	Natural Questions (NQ) is a large-scale open-domain question answering benchmark built from real user information-seeking queries issued to web search systems. Unlike synthetic or heavily curated QA datasets, NQ reflects authentic, noisy, and diverse question intents, making it well-suited for evaluating practical question answering performance in realistic settings. The dataset is grounded in Wikipedia documents and includes annotations for short and long answers, requiring models to identify both relevant evidence and precise answer spans. NQ emphasizes factual retrieval, answer extraction, and robustness to ambiguous or underspecified queries, and is widely used to assess whether language models can provide accurate and concise responses under real-world search-style distributions.
PopQA	Question Answering	PopQA is an open-domain question answering dataset designed to stress-test parametric factual knowledge under popularity-aware distributions. It contains subject-relation-object style factual questions derived from knowledge graph triples, with accompanying metadata such as entity popularity that helps characterize long-tail difficulty. PopQA is particularly useful for evaluating whether models can answer less frequent or less memorized facts, rather than only high-frequency popular entities. For model routing, this benchmark provides a practical way to compare robustness across the popularity spectrum and to determine whether cheaper models suffice for common facts while stronger models are needed for harder, low-frequency knowledge queries.
Musique	Question Answering	MuSiQue is a multi-hop question answering benchmark that explicitly targets compositional reasoning across multiple supporting paragraphs. Questions are built to require combining evidence from several facts, reducing shortcut opportunities that simpler single-hop datasets may allow. The benchmark includes decomposed reasoning supervision and supporting context structure, enabling analysis of both final-answer correctness and intermediate reasoning demands. In routing scenarios, MuSiQue is important because it distinguishes models that can perform deeper evidence composition and cross-document inference, helping select when high-capability models are necessary for complex reasoning-heavy QA requests.
Bamboogle	Question Answering	Bamboogle is a challenge-style open-domain question answering dataset curated to include difficult, often deceptive or compositionally tricky questions that can expose weaknesses in shallow retrieval-and-match behavior. Many items require careful factual disambiguation, multi-step inference, or resistance to plausible but incorrect distractor knowledge. Compared with standard factual QA sets, Bamboogle places higher emphasis on robustness under adversarially challenging query formulations. For routing systems, Bamboogle is a useful stress benchmark to evaluate whether the router can identify hard queries and assign them to models with stronger reasoning and calibration, instead of over-allocating to lightweight models.

A.7 Controls for Routing Pool Construction

Table 13: Alternative pool constructors on NQ under the Router-R1 protocol of Table 4, with the same candidate universe and pool size. **Recency**: each original slot is replaced by the most recent same-family release in the same scale bucket (HuggingFace release dates, dense models matched on parameter count, MoE models on active parameters); "kept" marks slots with no eligible newer release. **ZAP**: pool selected by the strongest feature-free baseline of Table 1, listed by scale. **MODELLENS**: the recommended pool of Table 3.

Original (Router-R1)	**Recency**	**ZAP**	**MODELLENS**
LLaMA-3.1-70B	Llama-3.3-70B	Llama-3.3-70B-Instruct	LLaMA-3.3-70B
Mixtral-8x22B	Mixtral-8x22B (kept)	Qwen2.5-32B-Instruct	GPT-OSS-20B
Gemma-2-27B	Gemma-3-27B-IT	GPT-OSS-20B	Llama-4-Maverick
LLaMA-3.1-8B	LLaMA-3.1-8B (kept)	Mixtral-8x7B-Instruct-v0.1	Nemotron-H-8B-R
Qwen2.5-7B	Qwen3.5-9B	Meta-Llama-3-8B-Instruct	Qwen2.5-7B
Mistral-7B	Ministral-8B-2410	Qwen2.5-7B-Instruct	Gemma-3n-E4B
NQ EM	0.508	0.464	**0.524**
Active params	≈161B	≈150B	≈**126**B
Original pool: NQ EM 0.388, ≈158B active parameters.			

A.8 Per-Task-Family Breakdown on New Datasets

Table 14 stratifies the 609 held-out datasets of Table 1 (New Datasets) by task family. MODELLENS improves over ZAP in every family, but the margin and the absolute ranking quality vary. It is strongest where candidate rankings follow transferable family- or scale-level regularities (speech recognition, bitext mining, classification), and weakest on embedding- and similarity-oriented tasks (reranking, retrieval, STS, pair classification), where rankings depend on narrow dataset-specific specialization that size and family priors do not capture; the same holds below $\sim$1B parameters, where the size prior is least reliable (Figure 2).

Table 14: Weighted Kendall τ_w per task family on the 609 held-out datasets, averaged over the datasets of each family; families with fewer than eight datasets are omitted. Bottom: fraction of held-out datasets on which the single best-performing model is ranked first, or within the top 10.

Task family	**#Datasets**	**MODELLENS**	**ZAP**	**Δ**
Speech recognition	25	0.832	0.053	+0.78
Bitext mining	116	0.781	0.150	+0.63
Classification	148	0.779	0.407	+0.37
Question answering	13	0.677	0.075	+0.60
Clustering	34	0.673	0.368	+0.31
Pair classification	13	0.625	0.342	+0.28
STS	56	0.615	0.375	+0.24
Text generation	84	0.602	0.099	+0.50
Retrieval	71	0.578	0.337	+0.24
Token classification	9	0.560	−0.331	+0.89
Reranking	14	0.499	0.251	+0.25
Best model at rank 1 (%)	609	**31.6**	0.9	+30.7
Best model in top 10 (%)	609	**59.7**	35.3	+24.4

Table 15: Model selection performance measured by MRR (for reference).

Method	Aircraft	Cars	DTD	Pets	Flowers102	Food101	Country211	EuroSAT	Avg.
Feature-based Transferability Methods									
H-Score	0.200	0.200	0.200	1.000	0.111	0.143	0.500	0.167	0.315
NCE	0.333	0.250	1.000	0.333	0.333	0.111	0.500	0.143	0.375
LEEP	0.250	1.000	0.500	0.200	0.500	0.167	0.333	0.111	0.383
NLEEP	0.100	0.111	0.100	0.333	0.333	0.100	0.500	0.111	0.211
LogME	0.125	0.111	0.100	0.500	0.125	0.200	0.500	0.167	0.229
PACTran	0.125	0.100	0.100	1.000	0.167	0.125	0.125	1.000	0.343
OTCE	0.200	0.100	0.100	1.000	0.143	0.111	0.125	1.000	0.347
LFC	0.143	0.200	0.100	0.250	0.333	0.333	0.200	0.500	0.257
GBC	0.100	0.100	0.333	0.200	0.167	0.333	0.333	0.111	0.210
Model-Spider	0.500	1.000	0.500	0.500	0.125	0.250	0.333	0.200	0.426
Know2Vec	0.250	0.250	0.333	1.000	0.167	0.111	0.500	0.111	0.340
Feature-free Methods									
Task2Vec	0.143	0.111	0.200	0.125	0.333	0.250	0.500	0.250	0.239
ZAP	0.100	0.200	1.000	0.111	0.100	1.000	0.250	0.200	0.370
Ours									
Ours (Feature Free)	0.250	0.333	1.000	0.250	0.250	0.500	1.000	0.333	0.490
Ours (Feature Aug.)	**0.500**	**1.000**	0.333	**1.000**	**0.333**	**1.000**	**0.500**	**0.500**	**0.635**

Table 16: Ablation study on different loss combinations. Best results are in bold.

Method	Loss	τ_w	NDCG@1	Hit@1	Recall@1	NDCG@10	Hit@10	Recall@10	NDCG@30	Hit@30	Recall@30
Full (ensemble)	L+P+Pt	**0.745**	**0.910**	**0.266**	**0.266**	**0.951**	**0.456**	**0.303**	**0.962**	**0.666**	**0.631**
NoPointWise	L+P	0.728	0.897	0.126	0.126	0.935	0.419	0.300	0.950	0.593	0.620
NoPairWise	L+Pt	0.703	0.896	0.080	0.080	0.930	0.405	0.284	0.942	0.631	0.582
NoListWise	P+Pt	0.632	0.892	0.223	0.223	0.906	0.322	0.241	0.912	0.475	0.410
OnlyPairWise	P	0.591	0.885	0.015	0.015	0.900	0.308	0.208	0.908	0.473	0.431
OnlyListWise	L	0.649	0.891	0.013	0.013	0.901	0.294	0.223	0.924	0.456	0.442

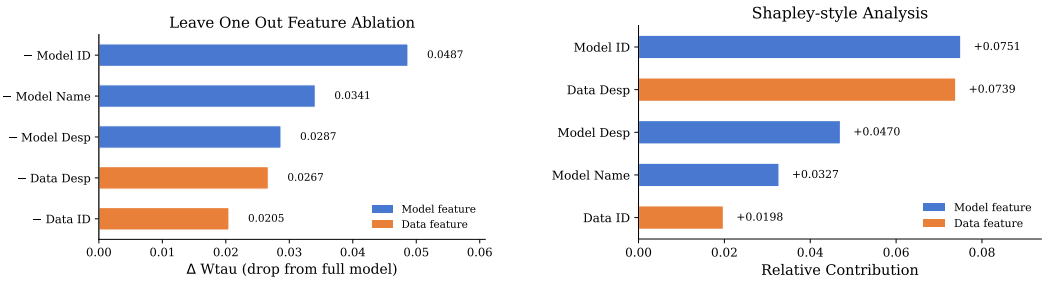

Figure 6: Comparison of ablation results and feature importance analysis.

A.9 Feature Ablation and Results

We investigate the contribution of model-side (model ID, name, description) and dataset-side (data ID, description) features using two complementary attribution views, as illustrated in Figure 6. LOO drops quantify the performance degradation when each feature is removed from the full model, capturing its contribution under a fixed context. In contrast, Shapley-style analysis [76] measures the average marginal contribution of each feature across feasible feature subsets, providing a context-agnostic attribution. Comparing LOO and Shapley estimates reveals strong feature interactions. In particular, Model Name and Model Desp exhibit pronounced redundancy: each contributes significantly in isolation, but their marginal gains shrink when combined, indicating overlapping model-identifying semantics that are obscured under LOO but captured by Shapley attribution. Overall, while LOO highlights feature importance in the presence of all signals, Shapley-style analysis uncovers complementary contributions and redundancy across features.

A.10 Unseen-Family Generalization

A practical model-recommendation system is most useful precisely when a *new* family of large language models appears: a user has a benchmark in mind, several Llama-, Qwen-, or Phi-class checkpoints have just been released, and the system is asked to rank them *before* any of those checkpoints have been re-evaluated on every benchmark. The two splits used elsewhere in the paper (random hold-out and held-out datasets) only test generalization within the same model population that the system was trained on. They cannot answer the harder question: *when an entire new family is held out from training, does the recommender still rank its members correctly?*

To probe this, we construct a **Modern-Cohort family hold-out** split (Section A.10.1) and evaluate three models on it (Section A.10.3).

A.10.1 Data Split

We define a *modern* cohort of 13 LLM families that drove most of the 2023–2025 open-weight progress: `qwen`, `llama`, `mistral`, `gemma`, `phi`, `deepseek`, `yi`, `falcon`, `granite`, `aya`, `olmo`, `zephyr`, `solar`. Every $(model, dataset, metric)$ row whose model belongs to one of these families is moved to the **test** split; rows whose model belongs to any other family form the **train** pool. Within the train pool we further carve out a 5% **model-disjoint validation** slice, so that early stopping is also driven by an out-of-distribution signal at the model level.

The held-out family identifiers remain in the global vocabulary (their embeddings exist but receive no gradient during training); this isolates the question we want to ask. Were they removed from the vocabulary entirely, the model would have no way at inference time to address the embedding slot of, say, a Qwen checkpoint, and would simply fall back to a default prior. Keeping the slot but starving it of supervision tests whether the rest of the architecture (size prior, description embeddings, dataset latents) can *compensate* for an uninformative family signal.

The resulting test split contains $364{,}517$ rows over $4{,}943$ unique models and $2{,}040$ datasets, expressed as $19{,}850$ unique $(\text{dataset}, \text{metric})$ ranking tasks.

A.10.2 Models Compared

We evaluate three checkpoints on the held-out test set:

- **Holdout-Family**: trained on the family-hold-out train split with the family embedding pathway enabled.
- **AllSeen (ceiling)**: the standard model from the main paper, trained on the full data including all modern families. Provides an in-distribution upper bound under identical architecture and global vocabulary.

A.10.3 Results

Table 17 summarises the two trained models on the family-hold-out test set. Tables 18 and 19 break the result down by held-out family and by dataset overlap with training, respectively. All NDCG@K, Hit@K, and Recall@K entries are averaged only over $(\text{dataset}, \text{metric})$ tasks with at least K candidate models; the weighted Kendall τ ($w\tau$) column is averaged over all tasks.

A.10.4 Discussion

Evidence against family-level memorization. If the recommender were primarily relying on memorized family-specific leaderboard statistics, performance would be expected to collapse once all modern families are removed from training. Instead, the relatively small degradation in NDCG@10, Hit@10, and Recall@10 indicates that the model recovers a substantial fraction of ranking quality from transferable signals beyond explicit family identity, including dataset semantics, model descriptions, scale priors, and latent interaction structure.

Top-K ranking quality is largely preserved. On NDCG@10 — the metric most relevant for a recommender that surfaces a short candidate list to the user — the gap between Holdout-Family and the AllSeen ceiling is only 0.037 (a 4.9% relative drop). Hit@10 is in fact 0.008 *higher* for the hold-out model (0.7751 vs 0.7672). This suggests that even when an entire family is excluded from

Table 17: Held-out modern-family generalization. **Holdout-Family** is trained without any modern-family rows; **AllSeen** is trained on the full data and serves as the in-distribution ceiling. NDCG, Hit, and Recall at K are averaged over the $(\text{dataset}, \text{metric})$ tasks with at least K candidate models. $w\tau$ is the size-weighted Kendall τ over all $19{,}850$ tasks. The NDCG@10 / Hit@10 / Recall@10 gaps are all within 5 percentage points (and Hit@10 is in fact higher under hold-out), confirming that the model recovers top-K ranking quality on entirely unseen LLM families.

Metric	Holdout-Family	AllSeen (ceiling)	Δ
NDCG@1	0.4485	0.5070	−0.0585
NDCG@5	0.5888	0.6299	−0.0411
NDCG@10	0.7231	0.7605	−0.0374
NDCG@30	0.7476	0.8491	−0.1015
NDCG@50	0.8531	0.9266	−0.0735
Hit@1	0.1597	0.2060	−0.0463
Hit@10	0.7751	0.7672	0.0079
Hit@50	0.6684	0.8481	−0.1797
Recall@1	0.1597	0.2060	−0.0463
Recall@10	0.7382	0.7830	−0.0448
Recall@50	0.7563	0.8230	−0.0667
$w\tau$	−0.0754	0.1573	−0.2327

Table 18: Per held-out family, NDCG@10 and weighted Kendall τ (test $(\text{dataset}, \text{metric})$ groups containing at least one model of the listed family). Families are sorted ascending by the Holdout-Family − AllSeen NDCG@10 gap (top of the table = best generalization). *Yi* and *granite* essentially close the gap to the in-distribution ceiling at NDCG@10 ($\Delta \leq 0.013$); *deepseek*, *olmo*, and *gemma* show the largest drops, indicating that their performance profile is the least similar to the non-modern training cohort.

Family	#groups	NDCG@10		Δ	$w\tau$	
		Holdout-Family	AllSeen	NDCG@10	Holdout-Family	AllSeen
yi	8,404	0.7366	0.7354	0.0012	−0.0991	0.0061
granite	11,654	0.7507	0.7638	−0.0131	−0.1255	0.1344
phi	1,501	0.7994	0.8702	−0.0708	0.1868	0.5918
falcon	61	0.5800	0.6529	−0.0729	0.2891	0.5789
zephyr	59	0.5661	0.6432	−0.0771	0.2771	0.5438
aya	43	0.5257	0.6040	−0.0783	0.2998	0.5477
llama	2,277	0.7863	0.8698	−0.0835	0.1348	0.5680
mistral	5,103	0.7578	0.8457	−0.0879	0.1338	0.3273
solar	87	0.5908	0.6872	−0.0964	0.2241	0.5729
qwen	3,262	0.6797	0.8049	−0.1252	0.0887	0.5023
gemma	2,725	0.6423	0.7832	−0.1409	0.0734	0.5003
olmo	195	0.6116	0.7727	−0.1611	0.3586	0.2443
deepseek	253	0.6070	0.7747	−0.1677	−0.0499	0.5916

training, the recommender's top-10 shortlist still contains the truly best candidate roughly as often as the in-distribution ceiling does.

Fine-grained ordering degrades. The gap is larger on the size-weighted Kendall τ (-0.075 vs 0.157). Without family-level supervision the model cannot fully resolve the ordering among near-equivalent modern checkpoints, but it does identify the right *set* of strong candidates. For a model-recommendation use case this is the desirable failure mode: recovering the correct candidate pool is typically more important than perfectly ordering highly similar checkpoints.

Per-family heterogeneity. The hold-out cost is not uniform. Families whose performance profile is similar to non-modern reference models — `yi` and `granite` — generalize almost for free (NDCG@10 within 0.013 of the ceiling). Families that introduced architectural or training-data shifts not represented in the non-modern cohort — `deepseek`, `olmo`, `gemma`, `qwen` — suffer the largest drops (ΔNDCG@10 between -0.13 and -0.17), indicating that some aspects of "family identity"

Table 19: Held-out family test set, split by whether the test $(\text{dataset}, \text{metric})$ key was also present in training (with non-modern models). *Seen-Dataset* measures pure model-side OOD; *Unseen-Dataset* measures model+dataset double OOD. NDCG@10 gaps are nearly identical in both regimes, suggesting that the family signal is disentangled from the dataset signal.

Bucket	#tasks	NDCG@1	NDCG@5	NDCG@10	NDCG@30	$w\tau$
Seen-Dataset (model-side OOD)						
Holdout-Family	13,106	0.4226	0.5921	0.7019	0.7479	−0.0428
AllSeen (ceiling)	13,106	0.4526	0.6303	0.7358	0.8485	0.1615
Unseen-Dataset (model+dataset OOD)						
Holdout-Family	6,744	0.4989	0.5845	0.7663	0.7149	−0.1387
AllSeen (ceiling)	6,744	0.6128	0.6293	0.8109	0.9027	0.1492

are not fully recoverable from size and description alone. This suggests that modern model families occupy partially distinct regions of the learned model–dataset interaction manifold.

Decoupling from dataset overlap. The NDCG@10 gap is nearly the same in the Seen-Dataset bucket (model-only OOD: -0.034) and the Unseen-Dataset bucket (model+dataset OOD: -0.045). The recommender's ability to handle a brand-new family therefore does not appear to depend strongly on prior exposure to the target benchmark. Instead, the dominant difficulty under family hold-out arises from missing family-level signals rather than from unfamiliar datasets, suggesting a partial disentanglement between model-side and dataset-side generalization.

A.11 Computing Standardized Advantage and Learned Priors

Let $\mathcal{D} = \{(t, d, \mu, m, v)\}$ denote the evaluation table, where each row corresponds to t, dataset d, metric μ, model m, and its score v. Both size-prior and family-prior figures consist of two curves: (i) a DATA curve summarizing empirical performance, and (ii) a PROBE curve reflecting the model's learned bias.

A.11.1 Standardized Advantage (DATA)

To compare performance across heterogeneous metrics, we standardize scores within each (t, d, μ) group.

Within-group normalization. For each group with at least two models, we compute z-scores:

$$z_{m,(t,d,\mu)} = \frac{v_{m,(t,d,\mu)} - \mu_{(t,d,\mu)}}{\sigma_{(t,d,\mu)}},$$

where μ and σ are the group mean and standard deviation. We apply clipping to reduce the effect of small-sample noise.

Per-model aggregation. Each model's overall score is computed as the average over all groups it appears in:

$$\bar{z}_m = \frac{1}{|\mathcal{G}(m)|} \sum_{(t,d,\mu)\in\mathcal{G}(m)} z_{m,(t,d,\mu)}.$$

Group-level advantage. We define the standardized advantage of a group (size bucket or family) as the mean of $\bar{z}_m$ over all models in that group:

$$\mathrm{adv}(g) = \mathrm{mean}_{m\in g}\, \bar{z}_m.$$

A positive value indicates that the group consistently outperforms the average model on the same evaluations. We discard groups with insufficient samples.

A.11.2 Learned Prior (PROBE)

To isolate the learned size and family effects, we probe the model's prior head, which depends only on size and family embeddings:

$$\phi(b, f) = W_2\, \mathrm{ReLU}\left(W_1\, [\mathbf{e}_b^{\text{size}} \,\|\, \mathbf{e}_f^{\text{fam}}]\right) + \mathbf{b}_2.$$

To analyze each factor independently, we marginalize the other:

- **Size prior:** $s^{\text{size}}(b) = \phi(b, \bar{\mathbf{e}}^{\text{fam}})$
- **Family prior:** $s^{\text{fam}}(f) = \phi(\bar{\mathbf{e}}^{\text{size}}, f)$

where $\bar{\mathbf{e}}$ denotes the empirical mean embedding. For visualization, PROBE values are standardized across bins.

For each figure, we report:

- Spearman ρ between DATA and the group coordinate;
- Spearman ρ between PROBE and the group coordinate;
- Linear slope of PROBE with respect to size (log-scale) or family rank;
- For family, η^2 as the variance explained by family identity.

All figures can be reproduced from logged evaluation results and trained checkpoints. We provide scripts that recompute standardized scores and regenerate all plots from raw data.

Table 20: Full model ranking and evaluation metrics used in the case study. We report BLEU-1, BLEU-4, ROUGE-L, and METEOR for evaluated models. Models without evaluation results are omitted for clarity.

Rank	Model	MODELLENS Score	BLEU-1	BLEU-4	ROUGE-L	METEOR
#1	ovis2 (8b)	11.308	42.90	5.17	24.31	31.65
#2	internvl3-8b	10.695	33.36	3.24	20.51	29.49
#3	qwen2-vl-7b-instruct	10.458	43.73	6.02	23.19	28.83
#4	internvl2.5-8b	9.646	41.96	5.61	23.65	28.67
#5	qwen2.5-vl-7b-instruct	9.314	42.59	4.74	23.90	28.39
#6	llava-next (7b)	9.039	36.76	3.97	22.60	22.99
#7	llava-1.5-7b	8.633	31.91	2.91	22.76	21.91
#12	blip-2	7.201	13.59	0.77	8.45	3.65

A.12 Limitations

Our framework relies on publicly available leaderboard evaluations, which may contain reporting bias toward popular model families and benchmark datasets. In addition, while our framework generalizes across domains, cross-modality recommendation remains challenging when leaderboard coverage is sparse. Our experiments primarily evaluate open-source models available on HuggingFace and public leaderboards, which may not fully represent proprietary or closed-source frontier systems.

Our temporal split evaluates newly released models without any update. New models and datasets can be scored immediately through the [UNK] pathway without retraining, but absorbing their accumulated records into learned ID embeddings requires periodic retraining; we do not study the update cadence or temporally adaptive evaluation that a deployed system would require.

A.13 Broader Impacts

This work aims to improve the scalability and accessibility of foundation model selection by reducing the need for exhaustive model evaluation and fine-tuning. Potential positive impacts include lowering computational costs, enabling more efficient deployment of open-source models, and improving access to foundation models for smaller organizations. However, our framework may inherit biases present in public leaderboards and benchmark datasets. Over-optimization toward benchmark performance may also encourage narrow evaluation practices. Future work should investigate fairness-aware recommendation and robustness across underrepresented domains.

A.14 Assets and Licenses

We use publicly available model metadata and benchmark results from:

- HuggingFace Hub

- Open LLM Leaderboard

- Papers with Code

All datasets and models remain subject to their original licenses and terms of use.

A.15 Reproducibility Statement

We provide implementation details, preprocessing procedures, hyperparameters, and evaluation settings necessary to reproduce our experiments.

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 accurately describe the scope and contributions of the paper, including large-scale model–dataset interaction modeling, cross-domain model recommendation, and generalization to unseen models and datasets. Experimental results and case studies support the stated claims.

   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: The paper discusses several limitations, including dependence on available leaderboard coverage, sparsity in model–dataset interaction matrices, and challenges in extending recommendations across incompatible modalities. These limitations are discussed in the limitations section Section A.12.

   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 contain formal theoretical results requiring proofs.

   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: The paper provides sufficient implementation details to reproduce the experiments, including dataset construction procedures, model metadata extraction, training objectives, evaluation protocols, and hyperparameter settings. Additional implementation details are provided in the appendix.

   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 anonymized code, processed leaderboard interaction data, and training scripts upon publication. The supplemental material includes instructions for reproducing the main experiments.

   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: The paper specifies the datasets, leaderboard sources, evaluation metrics, model filtering criteria, training objectives, hyperparameters, and evaluation settings necessary to understand and reproduce the experiments.

   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: [Yes]

   Justification: We report averaged results over multiple random seeds and provide standard deviations for the main experimental results. The reported variance reflects randomness from model initialization, training order, and sampled train/validation/test splits where applicable.

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 appendix reports the hardware configuration, GPU types, memory usage, training time, and approximate total compute required for the experiments.

   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 research complies with the NeurIPS Code of Ethics. All experiments use publicly available datasets, model metadata, and benchmark evaluations.

   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: The paper discusses both positive and negative societal impacts. Potential benefits include reducing computational waste in model selection and improving accessibility of foundation model deployment. Potential risks include reinforcing leaderboard biases and over-reliance on benchmark-driven 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 paper does not release high-risk generative models or sensitive datasets requiring special safeguards.

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 datasets, models, and leaderboard assets used in this work are publicly available and properly cited. We follow the licenses and terms of use associated with each resource, including HuggingFace-hosted models and public benchmark datasets.

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 paper introduces a curated large-scale model–dataset interaction benchmark derived from public leaderboard evaluations. Dataset construction details, preprocessing steps, and documentation are provided in the appendix and supplemental material.

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, user studies, human annotation, or experiments with human subjects.

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 research and therefore does not require IRB approval or equivalent review.

    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: [Yes]

    Justification: Large language models are used only as auxiliary tools for generating or summarizing model and dataset descriptions used as metadata features. They are not used to generate experimental results or replace human evaluation.

    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.