STARMAP: REPRESENTING LANGUAGE MODEL CAPABILITIES THROUGH EMBEDDING SPACE GEOMETRY

Anonymous authors
Paper under double-blind review

ABSTRACT

The rapidly growing ecosystem of Large Language Models (LLMs) makes it increasingly challenging to manage and utilize the vast and dynamic pool of models effectively. We propose STARMAP, a method that produces low-dimensional vector embeddings that compactly represent a language model's capabilities across queries. STARMAP is an attention-based approach that generates embeddings by a deterministic forward pass over query encodings and evaluation scores via an encoder model, enabling seamless incorporation of new models to the pool and refinement of existing model embeddings without having to perform any retraining. We additionally train a correctness predictor that uses model embeddings and query encodings to achieve competitive routing accuracy on unseen queries. Experiments show that STARMAP needs up to $4.8\times$ fewer query evaluation samples than baselines to produce informative and robust embeddings. Moreover, the learned embedding space is geometrically meaningful: proximity reflects model similarity, enabling a range of downstream applications including model comparison and clustering, model portfolio selection, and resilient proxies of unavailable models. We additionally provide theoretical results on efficiency and stability of STARMAP, validating our empirical observations.

1 INTRODUCTION

Large Language Models (LLMs) have been utilized for a wide range of tasks OpenAI (2024); Brown et al. (2020); Bommasani et al. (2021), including but not limited to question answering, code generation, reasoning, and domain-specific assistants Hu et al. (2022). Concurrently, the number of available LLMs has also grown exponentially, with hundreds of proprietary models and thousands of open-source models.[1] These models vary greatly in size, architecture, training data, and expertise Lin et al. (2021), offering opportunities to match inference queries to the best-suited model Ding et al. (2024); Ong et al. (2025). Utilizing and managing a vast pool of language models raises many practical challenges Horwitz et al. (2025); Zhao et al. (2025). How can we compare models across sizes and architectures Tay et al. (2022)? How can we map model abilities to find similarities or differences among them Kornblith et al. (2019)? How can we minimize redundancy in a model pool by retaining a small subset of models while preserving their aggregated abilities Chang et al. (2024)? How can we maintain a dynamic pool Chiang et al. (2024) where new models are frequently added and older models are deprecated? Addressing these questions necessitates a representation that makes model capabilities easy to summarize and compare at scale.

We propose a simple yet powerful idea: represent each model by a fixed-dimensional vector that summarizes its capabilities across queries. We refer to this representation as a *model embedding*.

Why are model embeddings useful? By embedding models in a shared vector space, we can compare and organize models using simple approaches like nearest-neighbor lookup Fix & Hodges (1989) and clustering Hartigan & Wong (1979). This supports *model discovery and organization* (e.g., finding similar models, grouping model families, and understanding joint capability profile and redundancy), *portfolio selection* (choosing a smaller set of models while covering all capabilities), and *monitoring* (for security purposes and tracking model drift). Model embeddings are also useful

[1] Presently, over 300K text-to-text models are open-sourced on huggingface.co/models

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

for *decision making at inference time*: they can support query routing by estimating which models are likely to perform well on a given query, allowing selection of models best suited for each query.

Information sources for model embeddings. To capture capabilities of language models, we require an information source that reflects model performance on queries. In the full information setting, model parameters can be directly used for generating such embeddings, but it is infeasible due to heterogeneous model sizes and architectures Zhao et al. (2025) which are often proprietary Grattafiori & Team (2024); Gemini (2025); OpenAI (2024) and accessible only via APIs. An alternative is to distill model capabilities Hinton et al. (2015) from output logit distributions of generated tokens, which can be problematic when there are different tokenizations used across models Xu et al. (2024). Instead, a practical way of capturing the capabilities of diverse models is by observing their generated responses to queries, which can be assigned performance scores (e.g., correctness value) by comparison with ground truth answer Hu et al. (2024) or by using LLM-as-a-judge Li et al. (2025). Concretely, for a text query x and model m, we observe a text response and compute a score $y^{(m)}(x)$, often simply the binary correctness value. For each model m, we aim to map the observed set of evaluations into a vector z_m which captures its capabilities.

Desiderata. The aforementioned use cases inform key requisites from an embedding generation approach: (i) **Black-box compatible:** rely only on queries and evaluation scores, without requiring access to model weights or internal activations. (ii) **Compatible with varying evaluations:** do not require all models to be evaluated on the same fixed query set to obtain comparable embeddings. (iii) **Sample efficient:** require only a few query evaluations to produce informative embeddings, which improve as more evaluations are collected. (iv) **Informative of capabilities:** support accurate performance prediction on unseen queries. (v) **Geometrically meaningful:** proximity under distance measures (e.g., cosine or Euclidean) should reflect model similarity.

1.1 Related works.

Parametric approaches. A common parametric design learns *per-model embeddings as trainable parameters* along with a correctness predictor trained on model–query evaluations. EmbedLLM (Zhuang et al., 2025) follows this general design, while IRT-Net (Chen et al., 2025) and JE-IRT (Yao et al., 2025) use an item response theory formulation for interpretable predictions (Hambleton et al., 1991). A related parametric approach Kashani et al. (2025) utilizes matrix factorization to represent models in query encoding space. Refer Appendix F for additional remarks.

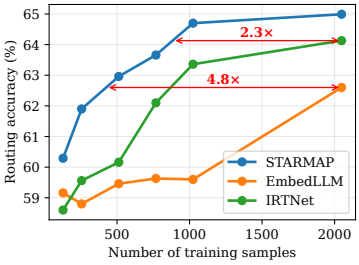


Figure 1: **Routing accuracy vs number of training samples.** STARMAP uses $\mathbf{2.3-4.8}\times$ fewer training samples (number of queries on which each model is evaluated) as compared to baselines, while maintaining high routing accuracy performance.

Nonparametric approaches. Nonparametric methods utilize common evaluation queries to directly compute *model embeddings using geometric operations*, without learning any parameters. LLM-DNA (Wu et al., 2025) concatenates response encodings and downprojects them using a Gaussian random matrix to obtain model embeddings. All models are required to be evaluated on the same queries, which does not generalize well to partial evaluations and evaluations on additional queries cannot be utilized for refining the embeddings.

1.2 Approach Overview and Contributions

Motivated by strengths of both parametric and nonparametric embedding generation approaches, we propose STARMAP, an attention-based method that processes a collection of queries evaluated on a model to generate a fixed-dimensional model embedding via an encoder F_θ. After training the encoder once, embedding for a model (including newly added models) is computed by a deterministic forward pass over its available evaluations, and the embedding can be further refined by adding more evaluations. In addition, we train a correctness predictor G_ψ which processes a model embedding and a query encoding to predict the correctness of model on that query.

The attention layers in STARMAP jointly process the available query evaluations, allowing them to extract a strong indicator of model capabilities even from limited data. We validate this insight

in Figure 1, where STARMAP achieves higher routing accuracy across a wide range of training-set sizes, with up to $\mathbf{4.8}\times$ **better sample efficiency** than prior baselines. The embeddings are less prone to overfitting to a particular model's evaluation set and remain stable as evaluation queries vary. The rich geometry of the embedding space enables several downstream tasks, including identifying model families, selecting model portfolios, and allowing resilient fallback decisions. We also provide theoretical results that deepen understanding of STARMAP, and support our empirical observations. The remainder of the paper is organized as follows: Section 2 introduces STARMAP; Section 3 presents theoretical results; Section 4 reports results and analyses; and Section 5 concludes our work.

2 USING ATTENTION FOR MODEL CAPABILITY REPRESENTATION

In this section, we describe an attention-based encoder (F_θ) for generating model embeddings from scored query evaluations, together with a query correctness predictor (G_ψ) that estimates a model's probability of correctness on a test query. Each query x is represented by a fixed dimensional encoding/embedding $\phi(x) \in \mathbb{R}^{d_\phi}$ (e.g., generated by a pretrained sentence encoder Wang et al. (2020); Lan et al. (2020)). Let $\mathcal{Q}_m$ denote the set of queries model $m \in \mathcal{M}$ is evaluated on. Let normalized evaluation scores be $y^{(m)}(x_i) \in [0, 1]$ for all queries $x_i \in \mathcal{Q}_m$. The evaluation set for model m is thus (with $n_m := |S_m|$ denoting the number of evaluations for model m):

$$S_m \;=\; \{(\phi(x_i),\, y^{(m)}(x_i))\}_{x_i \in \mathcal{Q}_m}.$$

In Section 2.1, we describe a tokenization scheme for individual evaluations, present the standard and an efficient variant of multi-head attention over evaluation tokens, and a final aggregation layer which produces model embeddings. Section 2.2 describes query correctness predictor, and Section 2.3 elaborates on the training procedure for (F_θ,G_ψ).

2.1 ATTENTION-BASED MODEL CAPABILITY ENCODER F_θ

Continuous tokenization of evaluations. We convert each evaluation pair $(\phi(x_i), y^{(m)}(x_i))$ into a token $t_i^{(m)}$ using a trained MLP $h_\omega : \mathbb{R}^{d_\phi} \times \mathbb{R} \to \mathbb{R}^{1\times d}$ as: $t_i^{(m)} = h_\omega\big(\phi(x_i), y^{(m)}(x_i)\big) \in \mathbb{R}^{1\times d}$, and stack tokens as rows to form

$$X_m^{(0)} \;=\; [(t_1^{(m)})^\top \cdots (t_{n_m}^{(m)})^\top]^\top \in \mathbb{R}^{n_m \times d}. \tag{1}$$

Intuitively, h_ω acts as a continuous tokenizer mapping heterogeneous inputs (query features + score) into a common token space suitable for attention operations. We view the tokenizer h_ω as the first stage of the encoder and define $X_m^{(0)} = h_\omega(S_m)$. The subsequent attention layers then map $X_m^{(0)} \mapsto z_m$, and by slight abuse of notation we write $F_\theta(S_m) \equiv F_\theta(X_m^{(0)})$. The encoder F_θ ultimately generates a single model embedding $z_m \in \mathbb{R}^d$ from the query evaluations.

We require the model embeddings to depend only on the query evaluation set S_m, and not on the particular order in which they are listed in $X_m^{(0)}$. To achieve this, we apply *bidirectional* attention *without* positional encodings. Under this design choice, shuffling the rows of $X_m^{(0)}$ only shuffles the intermediate token representations, implying permutation equivariance. We finally apply a learned-query attention aggregation step (eq. (3)) which maps the last layer's output tokens into a single model embedding vector z_m. This operation performs aggregation without any position specific weights, resulting in a final embedding that is permutation invariant to token ordering in $X_m^{(0)}$ (eq. (1)).

Attention over evaluation tokens. We denote a Transformer block Vaswani et al. (2017) by $\mathrm{TBlock}(\cdot,\cdot,\cdot)$, consisting of (i) a Multi-Head Attention (MHA) sublayer with a residual connection and layer normalization (LN_0), followed by (ii) a position-wise feed-forward network (FFN, a single hidden layer MLP $d \to d_{\mathrm{ff}} \to d$) applied independently to each token along with another residual connection and layer normalization (LN_1):

$$\bar{Q} = \mathrm{LN}_0\Big(Q + \mathrm{MHA}(Q,K,V)\Big) \in \mathbb{R}^{n_q\times d},\;\; \mathrm{TBlock}(Q,K,V) = \mathrm{LN}_1\Big(\bar{Q} + \mathrm{FFN}(\bar{Q})\Big) \in \mathbb{R}^{n_q\times d}.$$

Let $X_m^{(\ell)} \in \mathbb{R}^{n_m\times d}$ denote the output tokens after ℓ layers of Transformer blocks ($X_m^{(0)} \in \mathbb{R}^{n_m\times d}$ are input evaluation tokens in eq. (1)). Our self-attention block at layer ℓ over evaluation tokens is thus:

$$X_m^{(\ell)} \;=\; \mathrm{TBlock}\big(X_m^{(\ell-1)},\, X_m^{(\ell-1)},\, X_m^{(\ell-1)}\big) \in \mathbb{R}^{n_m\times d}.$$

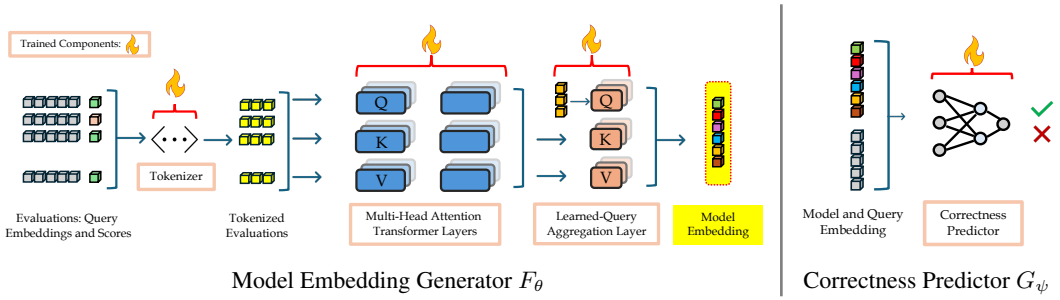


Figure 2: **Overview of STARMAP.** We design a model embedding generator and a correctness predictor which enables compact model representations and their capability predictions on queries. **(a) Model Embedding Generator.** Query evaluations for any language model are tokenized using the corresponding query encodings and correctness scores. Multi-Head Attention Transformer layers utilize bidirectional attention without positional encodings to capture information from the evaluations, which is finally captured by a learned query aggregation block to produce a model embedding. **(b) Correctness Predictor.** Model embedding and test query embedding are utilized by an MLP based decoder network to predict correctness probability of chosen model on the query.

Latent bottleneck attention block. Direct self-attention over n_m tokens incurs $\mathcal{O}(n_m^2)$ computational cost, which can be restrictive with large number of evaluations n_m. We reduce this by introducing $r \ll n_m$ learned latent vectors $U^{(\ell)} \in \mathbb{R}^{r \times d}$ which act as an attention bottleneck in each layer. Concretely, *one latent bottleneck attention block consists of two Transformer blocks*: a *compression* block where $U^{(\ell)}$ attends to the evaluation tokens $X_m^{(\ell-1)}$, followed by a *broadcast* block in which the evaluation tokens attend to the latent outputs $H_m^{(\ell)}$:

$$\boxed{\begin{aligned} H_m^{(\ell)} &= \text{TBlock}\big(U^{(\ell)},\ X_m^{(\ell-1)},\ X_m^{(\ell-1)}\big) \in \mathbb{R}^{r \times d}, \\ X_m^{(\ell)} &= \text{TBlock}\big(X_m^{(\ell-1)},\ H_m^{(\ell)},\ H_m^{(\ell)}\big) \in \mathbb{R}^{n_m \times d}. \end{aligned}} \tag{2}$$

This reduces the overall attention computation cost to $\mathcal{O}(n_m \times r)$ while still enabling full information flow through the attention layers Lee et al. (2019); Locatello et al. (2020); Jaegle et al. (2021).

Learned-query attention aggregation. After L latent-bottleneck attention blocks, we aggregate the n_m output token features into a single model embedding z_m using a trained query parameter $s \in \mathbb{R}^{1 \times d}$, which is learned with other model parameters during training:

$$\boxed{z_m = \text{TBlock}(s,\ X_m^{(L)},\ X_m^{(L)}) \in \mathbb{R}^{1 \times d}.} \tag{3}$$

This operation performs *attention-based aggregation*, where a fixed, learnable query attends over a variable number of token features Yang et al. (2016); Lin et al. (2017); Ilse et al. (2018); Lee et al. (2019), generating a single output vector which we denote as the model embedding. As this operation depends only on attention over the token features and does not use positional encodings, the resulting z_m is invariant to permutations of the input evaluations $X_m^{(0)}$ in eq. (1).

2.2 QUERY CORRECTNESS PREDICTOR G_ψ

Generated model embeddings $\{z_m\}_{m \in \mathcal{M}}$ in eq. (3) are actionable for model selection if we have a mechanism for translating the encoded information into correctness prediction of models for unseen queries. Such query-wise correctness prediction scores for language models can be used for a multitude of applications such as routing, confidence-based majority averaging of responses, ranking model performances etc. To this end, we design a correctness predictor (decoder) that processes query encodings along with model embedding vectors to return a probability of correctness value. Formally, given model embedding z_m and query encoding $\phi(x)$, the decoder predicts

$$\widehat{p}_\psi\left(y^{(m)}(x) = 1 \middle| z_m, \phi(x)\right) = \sigma\big(G_\psi\left(z_m, \phi(x)\right)\big), \tag{4}$$

where G_ψ is a lightweight MLP producing a scalar logit and $\sigma(\cdot)$ is the sigmoid function.

2.3 TRAINING AND USAGE

We train the model encoder F_θ (including tokenizer h_ω) and decoder G_ψ (correctness predictor) jointly from a supervised dataset containing model-query evaluation pairs and observed correctness values. Each step samples a mini-batch of models; for each model m, we form (i) an *encoder input subset* S_m^{enc} of scored evaluations to build z_m, and (ii) a *decoder query batch* S_m^{dec} on which we apply the correctness prediction loss. Additional details provided in Appendix H.

Batch structure and Training Loss. For any model m, let $S_m^{\text{enc}} \subseteq S_m$ be a randomly chosen subset of query evaluations, and let $z_m = F_\theta(S_m^{\text{enc}})$ denote the corresponding model embedding generated. Independently, we also randomly sample decoder queries $S_m^{\text{dec}} \subseteq S_m$ (with target labels $y^{(m)}(x)$ for $x \in S_m^{\text{dec}}$) and compute correctness probabilities in eq. (4) as $\widehat{p}^{(m)}(x) \triangleq \widehat{p}_\psi\Big(y^{(m)}(x) = 1 \Big| z_m, \phi(x)\Big)$. We minimize binary cross-entropy loss averaged over the models in a mini-batch and decoder queries:

$$\min_{\omega,\theta,\psi} \; \mathbb{E}_m \, \mathbb{E}_{x \sim S_m} \, \text{BCE}\Big(\widehat{p}^{(m)}(x),\, y^{(m)}(x)\Big) \, .$$

Generating embeddings for new models. To embed a new model m_{new}, we collect an evaluation set $S_{m_{\text{new}}}^{\text{enc}}$, and perform a forward pass over the embedding generator to obtain $z_{m_{\text{new}}} = F_\theta(S_{m_{\text{new}}}^{\text{enc}})$. No retraining or learning any additional parameters is required at embedding time.

Correctness prediction and routing. We estimate correctness probabilities $\widehat{p}^{(m)}(x)$ for candidate models and query x by using the correctness predictor G_ψ. These probabilities can be used directly for ranking models, routing to a suitable model for response generation, etc.

3 THEORETICAL ANALYSIS

We present theoretical results on three aspects of STARMAP: (i) quantifying the stability of model embeddings under varying query evaluations, (ii) demonstrating the robustness of routing accuracy and correctness predictions under perturbations in model embeddings (due to varying query evaluations), and (iii) examining the generalizability of STARMAP to previously unseen models. Detailed assumptions and proofs are deferred to Appendix I.

Given a model m with n_m available query evaluations $\mathcal{S}_m$, let $z_m = F_\theta(\mathcal{S}_m)$ denote its full-data embedding. For a subset $\widehat{\mathcal{S}}_{m,k} \subset \mathcal{S}_m$ of $k < n_m$ chosen evaluations sampled uniformly without replacement, the corresponding model embedding generated is $\widehat{z}_{m,k} = F_\theta(\widehat{\mathcal{S}}_{m,k})$. Under the boundedness and encoder-stability conditions (Appendix I), the embedding computed from a random subset concentrates around the full-data embedding as k increases.

Theorem 3.1 (Embedding stability under evaluation subsampling)**.** *For a given encoder, let k evaluations be sampled uniformly without replacement from $\mathcal{S}_m$. With probability at least $1 - \delta$,*

$$\|\widehat{z}_{m,k} - z_m\|_2 \leq \varepsilon_{m,k}(\delta),$$

where $\varepsilon_{m,k}(\delta)$ is the subsampling error bound (theorem I.4), and $\rho_{n_m,k}$ is the correction term for sampling without replacement. Furthermore, $\varepsilon_{m,k}(\delta)$ scales as

$$\varepsilon_{m,k}(\delta) = \widetilde{\mathcal{O}}_\theta\left(\sqrt{\frac{\rho_{n_m,k}}{k}}\right).$$

Corollary 3.2 (Sufficient query evaluations for routing stability)**.** *Assuming L_G-Lipschitz continuity of decoder G_ψ, the choice of the routed model for query x is preserved w.p. at least $1 - \delta$ when*

$$\gamma(x) > 2L_G \varepsilon_k(\delta),$$

where $\gamma(x) = G_\psi(z_{m^*(x)}, \phi(x)) - \max_{j \neq m^*(x)} G_\psi(z_j, \phi(x))$ denotes the routing margin between the best and the next best model, and $\varepsilon_k(\delta) = \max_{m \in \mathcal{M}} \varepsilon_{m,k}(\delta/|\mathcal{M}|)$.

Theorem 3.1 and corollary 3.2 justify the query evaluation sample efficiency for generating model embeddings and their relative robustness to varying query evaluations as observed in Sections 4.2 and 4.3 and Figures 4 and 5. For a previously unseen model m, STARMAP constructs an embedding from its evaluations by performing a forward pass through the trained encoder. Let $R_k(m)$ be the

expected decoder (correctness predictor) loss by utilizing a model embedding generated through k evaluations, and $R_{\text{full}}(m)$ be the corresponding loss using all n_m available evaluations, and $R^*(m)$ an optimal empirical risk value.

Theorem 3.3 (Risk decomposition for a new model)**.** *For binary cross-entropy loss, under the conditions of theorem 3.1 and the decoder Lipschitz continuity, with probability at least* $1-\delta$*,*

$$R_k(m) - R^*(m) \leq \underbrace{R_{\text{full}}(m) - R^*(m)}_{\textit{residual full-data error}} + L_G \varepsilon_{m,k}(\delta).$$

The first term captures the error between the full data embedding and the optimal empirical risk value, and the second term bounds the additional error from subsampling the query evaluations used to generate the model embedding. Training the encoder on more models and query evaluations is shown to reduce the residual full-data error by improving the shared encoder and correctness predictor representations, as observed in our evaluation in Section 4.2 and Figure 3, and examined in Tripuraneni et al. (2020); Maurer et al. (2016). We also extend STARMAP to online feedback scenario where query evaluations are sequentially presented through non-uniform sampling in Appendix I.5.

4 EXPERIMENTS

We evaluate our attention-based model embedding generator and query correctness predictor on five distinct aspects: (i) *Correctness prediction and routing accuracy* (ii) *Sample Efficiency* during training and test time (iii) *Robustness* of generated embeddings to changes in the evaluation set queries (iv) *Correlation* between embedding distance and correctness similarity of models (v) *Practical utilities* enabled by model embeddings.

Models, tasks, and evaluation sets. In our experiments, we consider a set of 112 language models Zhuang et al. (2025) spanning a wide variety of sizes and capabilities, including both general and finetuned models. The queries in our analysis are sampled from 10 public benchmarks (with binary scoring), and represent a real-world corpus across a variety of tasks such as math, general knowledge, reasoning etc. We also examine STARMAP on query responses with continuous evaluation scores in Appendix B.1. Additional details on language models used and query datasets are provided in Appendix A. We select the sentence encoder `all-mpnet-base-v2` Song et al. (2020) as $\phi(x)$ for generating query representations in our analysis, and defer ablations on other sentence encoders to Appendix C.

Our attention-based encoder[2] has two latent bottleneck attention blocks (eq. (2)) ($L = 2$), 4-headed MHA (Multi-Head Attention) sublayers, $r = 64$ learned latent vectors in each block, followed by attention-based aggregation layer (eq. (3)). The correctness predictor (eq. (4)) is a single hidden-layer MLP ($d_{\text{hidden}} = 64$). The model embedding dimension is set to $d = 128$ (results on varying hyperparameter choices is deferred to Appendix D). We do not utilize any test queries for generating model embeddings; they are only used for evaluating correctness prediction and performance agreement rate of models (Further details in Appendix H).

Table 1: Overall routing and correctness prediction accuracy (%) across approaches and training-set sizes. **Best** and second best emphasized.

Approach (# Query Evaluations)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
STARMAP (Ours)	**61.90**	**62.97**	**64.70**	**68.31**	68.33	70.03
EMBEDLLM	58.80	59.47	59.60	67.33	68.12	69.47
IRT-NET	59.57	60.17	63.37	67.38	**69.07**	**70.12**

Baselines. We compare against two baselines that obtain model embeddings via gradient-based training from query evaluation data: **EmbedLLM** Zhuang et al. (2025), and **IRT-Net** Chen et al. (2025). We defer comparison with nonparametric approaches (LLM-DNA Wu et al. (2025) and kNN router Jitkrittum et al. (2025)) to Appendix B.2.

4.1 CORRECTNESS PREDICTION AND ROUTING ACCURACY

For each model m and query x, the correctness predictor estimates $\widehat{p}^{(m)}(x) = \sigma(G_\psi(z_m, \phi(x))) \in [0,1]$. We convert the probability estimate into a binary prediction $\widehat{y}^{(m)}(x) = \mathbb{I}\{\widehat{p}^{(m)}(x) \geq 0.5\}$ by thresholding, and then average the agreement indicator $\mathbb{I}\{\widehat{y}^{(m)}(x) = y^{(m)}(x)\}$ over all model-query

[2]trained on 4096 query evaluations for each language model, unless mentioned otherwise.

pairs to calculate correctness prediction accuracy. Additionally, we judge whether the language model with highest predicted correctness probability $\widehat{p}^{(m)}(x)$ actually generated a correct response to query x, and report an average over all test queries as the overall routing accuracy.

Across training set sizes, STARMAP outperforms examined model embedding baselines in overall routing accuracy, and closely matches correctness prediction performance as presented in Figure 1 and Table 1. We attribute this to the ability of attention blocks in extracting model capability patterns through query evaluations, instead of using backpropagation to obtain model embeddings that describes observed correctness patterns. We extend further examination on numerical performance in Appendices B to D, and wall clock time complexity in Appendix G.

4.2 SAMPLE EFFICIENCY

Training-time sample efficiency. Figure 1 reports routing accuracy as a function of the number of query evaluations per model used to train (F_θ, G_ψ) (i.e., $n_m = |S_m|$). Our method reaches strong correctness prediction and routing performance with substantially fewer training evaluations, resulting in **upto $4.8\times$ sample efficiency** as compared to existing baselines. Notably, our training procedure does not require all models to share the same evaluation queries, nor to be evaluated on the same number of queries, further increasing practicality of our approach.

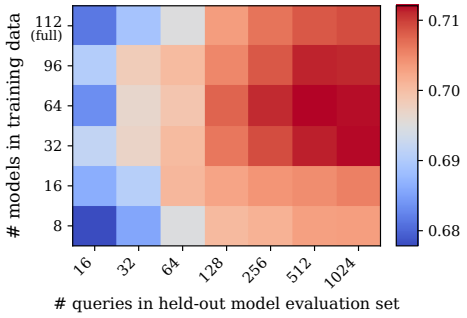


Figure 3: **New LLM onboarding.** Correctness prediction accuracy on 16 held-out models: with varying number of query evaluations for embedding held-out models and varying number of models present in training data (top row indicates (F_θ, G_ψ) trained on all language models).

Test-time embedding efficiency (new model onboarding). To measure sample efficiency for generating embeddings for unseen models, we train (F_θ, G_ψ) on query evaluation data from # models $\in \{8, 16, \ldots, 96\}$ (1024 evaluations each) and then embed 16 held-out models using # query evaluations $\in \{16, 32, \ldots, 1024\}$ (details in Appendix A). Figure 3 shows that as few as $\approx$128 evaluations are sufficient to obtain informative embeddings. We also observe negligible gap ($< 1\%$) between correctness prediction accuracy using partial data trained versus full data trained (F_θ, G_ψ), highlighting generalization to unseen models.

4.3 ROBUSTNESS TO VARYING QUERY EVALUATIONS

Robustness to the choice of evaluation queries. We test how sensitive model embeddings are to the query evaluation set used as input to the encoder. Using a pretrained encoder F_θ trained from evaluations on the full model pool, we first compute *reference embeddings* z_m^{ref} for each model m from a fixed reference query evaluation set of size n_{ref} (e.g., 4096). We then perturb the input evaluations in two ways: changing **which** queries are used (fixed number but different queries) and changing **how many** queries are used (subsampling from reference evaluation set), and then analyze the stability of embeddings via routing/correctness prediction and embedding geometry.

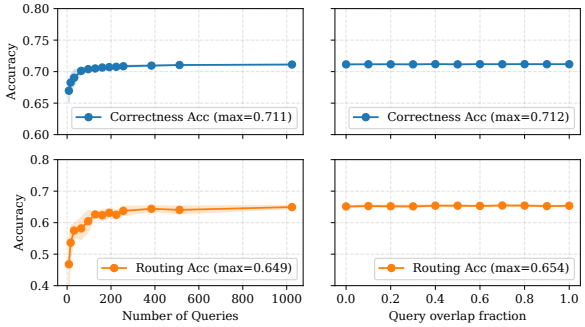


Figure 4: **Robustness to Varying Query Evaluations.** Accuracy for correctness prediction (top) and routing (bottom) as a function of (left) the number of evaluations used to construct embeddings and (right) the overlap fraction with a reference evaluation set.

Keeping the number of queries in the evaluation set fixed at n_{ref}, we resample alternate evaluation sets whose overlap with the reference set is $\alpha \in [0, 1]$. We then recompute embeddings for all models, and calculate numerical metrics using the new embeddings. Figure 4 shows that correctness prediction and routing accuracy remain stable across overlap fractions, and Figure 5 illustrates that the regenerated embeddings stay close to originally generated embeddings z_m^{ref} (downprojection via t-SNE van der Maaten & Hinton (2008)).

Robustness to the number of evaluation queries. We also subsample n evaluations from the reference set of query evaluations (e.g., $n \in \{64, 128, 256, \ldots, n_{\text{ref}}\}$) and recompute embeddings. Numerical performance saturates quickly: embeddings formed from only ≈ 128–256 evaluations typically match the routing and correctness prediction accuracy obtained with larger sets (Figure 4). The corresponding t-SNE projections of embeddings in Figure 5 further suggest geometric stability under subsampling evaluation queries.

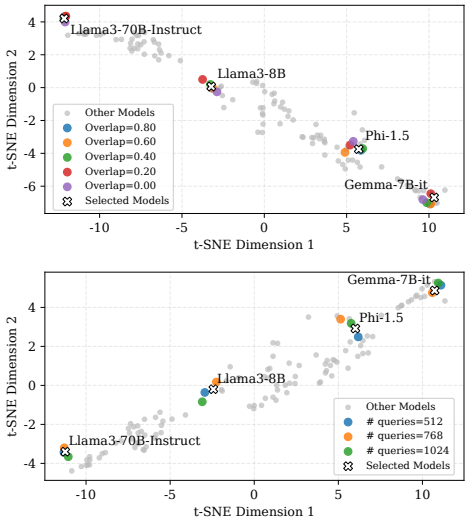


Figure 5: **Robustness to varying query evaluations.** t-SNE embeddings for selected models under varying overlap with the reference set and subsampling from reference set. Gray points denote other models.

Implications for model fingerprinting (security). Robustness to query evaluations provides a preliminary fingerprinting signal: if a new model's embedding repeatedly maps near the same reference model across independently sampled evaluation sets, it suggests potential model duplication (refer Appendix E.3 for additional experiments) Zeng et al. (2024); Tsai et al. (2025).

4.4 Embedding Geometry Reflects Similarity

We next test whether distances in the learned embedding space track differences between model capabilities. Concretely, for every model pair $(m_1, m_2) \in \mathcal{M} \times \mathcal{M}$ we compute (i) an embedding-space distance $d_z(z_{m_1}, z_{m_2})$ (cosine and Euclidean), and (ii) their correctness *disagreement rate* on a common test query set, defined as the fraction of queries on which their binary correctness labels differ (i.e., the normalized Hamming distance, or $1-$agreement). Figure 6 shows a strong positive relationship between embedding distance and disagreement across all correlation measures: for example, Pearson correlation is 0.845 (cosine) and 0.887 (Euclidean), with high rank correlations (Spearman ≈ 0.88 and Kendall ≈ 0.71). These results indicate that embedding distances convey meaningful information about model similarity.

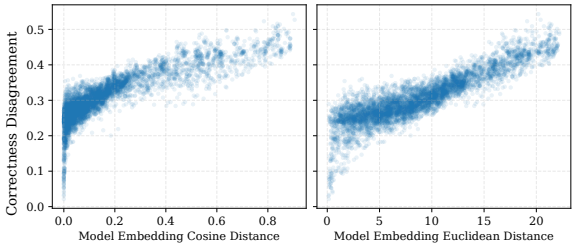


Figure 6: **Embedding distance versus correctness disagreement.** Embedding distances correlate strongly with disagreement in observed correctness outcomes, indicating that distances among embeddings express model similarity.

Recovering model specializations via hierarchical clustering. We perform hierarchical clustering Cohen-addad et al. (2019) using pairwise distances (Euclidean distance) between embeddings. The dendrogram and distance heatmap in Figure 7 reveal coherent groupings that align with known model specializations (e.g., math- and code-finetuned models), suggesting that the embedding geometry also captures shared capability structure at the level of model specialization rather than individual models.

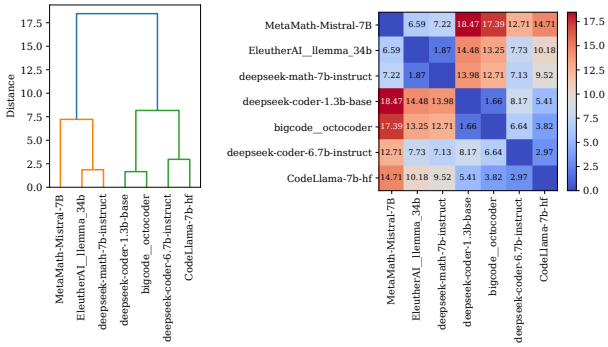


Figure 7: **Identifying model families.** Hierarchical clustering on model embeddings meaningfully separates model families, illustrated here for math (orange) and code (green).

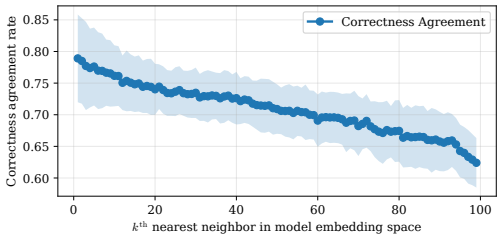


Figure 8: **Nearest-neighbor model proxies.** Average correctness agreement between a model and its k-th nearest neighbor in embedding space (averaged over models).

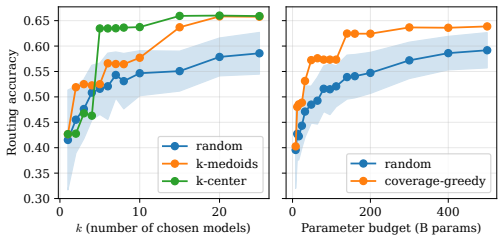


Figure 9: **Model Portfolio Selection.** Using embeddings to choose a model subset that maximizes routing accuracy. *Left (count-constrained)*, *Right (parameter-budget)*.

4.5 Practical utilities enabled by model embeddings

Nearest neighbors as proxies under model unavailability. In practice, a preferred model may be unavailable due to capacity limits, hardware failures, policy constraints, or other factors. We therefore test whether nearest neighbors in embedding space can serve as reliable *proxy* models with similar capabilities. For each model m, we rank neighbors by embedding distance, and measure correctness agreement on a shared query set. Figure 8 reports average agreement for the kth neighbor across all models, with the closest neighbor reaching **79%** correctness agreement and agreement decreasing gradually for larger k.

Model portfolio selection from embeddings. We evaluate whether the *model embedding space* can guide the selection of a *portfolio* of models that preserves routing accuracy relative to the full pool by reducing redundancy in capabilities across model.

Count-constrained portfolios. Given a budget of k models, we select diverse portfolios by maximizing coverage in embedding space using `k-center` (minimizes the maximum distance between all models' embeddings to those in the portfolio) Gonzalez (1985) and `k-medoids` (minimize average distance of all models' embeddings to models in the portfolio) Kaufman & Rousseeuw (1990). We observe that a **15-model portfolio captures almost full routing accuracy** of 112 models in the pool (details deferred to Appendix E.4.2). Figure 9 reports routing accuracy restricted to models in the portfolio as k varies.

Parameter-budget portfolios. Given a total parameter budget, the (`coverage-greedy`) algorithm selects a model subset that covers the model embedding space while satisfying the parameter budget (Appendix E.4.3). Figure 9 depicts strong routing performance with relatively small parameter budget (full pool of 112 models has a total of 1930B parameters).

Additional Utilities. Model embeddings also enable resilient routing and retrieval of the best model for a desired capability profile. More details are presented in Appendix E.

5 Concluding Remarks

In this paper, we present STARMAP, a novel attention-based framework that projects Large Language Models into a low-dimensional embedding space based on their inference capabilities. Our work demonstrates superior sample efficiency, requiring up to $4.8\times$ fewer query evaluations to generate robust representations. Beyond standard performance prediction, we show that the model embedding geometry captures behavioral similarity, enabling diverse downstream utilities such as efficient model portfolio selection and resilient fallback routing under model unavailability. As the ecosystem of proprietary and open-source models continues to expand, such capability-oriented embeddings offer a scalable, geometrically meaningful foundation for efficient model management and discovery. Extensions to embeddings for multimodal models is a promising future direction.

AI use statement

Authors have used AI tools to polish human-written text presented in the paper, along with finding related work to the presented directions. Authors have used AI assistance for clear presentation of figures and plots, and for initial experimentation design. Authors have not used AI assistance in any form besides already mentioned. Authors take responsibility of the accuracy and validity of the ideas and claims presented in this paper.

Ethics Statement

The authors acknowledge and adhere to the ICLR Code of Ethics at all stages of the conference submission, peer review and publishing process.

Reproducibility Statement

All experiments presented in this paper are reproducible via the code present in the supplementary submission. Complete information required for verifying the main experiments and claims of the paper is provided in Appendix H. The hyperparameter choices utilized across the paper are mentioned in Section 4, and ablations presented in Appendix D.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1245. URL https://aclanthology.org/N19-1245.

Rémi Bardenet and Odalric-Ambrym Maillard. Concentration inequalities for sampling without replacement. *Bernoulli*, 21(3):1361–1385, 2015.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, Apr. 2020. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Rishi Bommasani et al. On the opportunities and risks of foundation models. *ArXiv*, 2021. URL https://crfm.stanford.edu/assets/report.pdf.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), March 2024. ISSN 2157-6904. doi: 10.1145/3641289. URL https://doi.org/10.1145/3641289.

Jianhao Chen, Chenxu Wang, Gengrui Zhang, Peng Ye, Lei Bai, Wei Hu, Yuzhong Qu, and Shuyue Hu. Learning compact representations of llm abilities via item response theory, 2025. URL https://arxiv.org/abs/2510.00844.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference. *CoRR*, abs/2403.04132, 2024. URL https://doi.org/10.48550/arXiv.2403.04132.

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

Vincent Cohen-addad, Varun Kanade, Frederik Mallmann-trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *J. ACM*, 66(4), June 2019. ISSN 0004-5411. doi: 10.1145/3321386. URL https://doi.org/10.1145/3321386.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Evelyn Fix and J. L. Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review / Revue Internationale de Statistique*, 57(3):238–247, 1989. ISSN 03067734, 17515823. URL http://www.jstor.org/stable/1403797.

Team Gemini. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(85)90224-5. URL https://www.sciencedirect.com/science/article/pii/0304397585902245.

Aaron Grattafiori and Meta Team. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Ronald K. Hambleton, Hariharan Swaminathan, and H. Jane Rogers. *Fundamentals of Item Response Theory*, volume 2 of *Measurement Methods for the Social Sciences*. SAGE Publications, Newbury Park, CA, 1991. ISBN 0-8039-3647-8.

J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2346830.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Eliahu Horwitz, Nitzan Kurer, Jonathan Kahana, Liel Amar, and Yedid Hoshen. We should chart an atlas of all the world's models. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems Position Paper Track*, 2025. URL https://openreview.net/forum?id=BzFMBNqg7R.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

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. In *Agentic Markets Workshop at ICML 2024*, 2024. URL https://openreview.net/forum?id=IVXmV8Uxwh.

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2127–2136. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/ilse18a.html.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021. URL https://arxiv.org/abs/2103.03206.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient llm inference, 2025. URL https://arxiv.org/abs/2502.08773.

Idan Kashani, Avi Mendelson, and Yaniv Nemcovsky. Representing LLMs in prompt semantic task space. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 8578–8597, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.456. URL https://aclanthology.org/2025.findings-emnlp.456/.

Leonard Kaufman and Peter J. Rousseeuw. Partitioning around medoids (program pam). In *Finding Groups in Data: An Introduction to Cluster Analysis*, chapter 2, pp. 68–125. John Wiley & Sons, Inc., 1990. ISBN 9780470316801. doi: 10.1002/9780470316801.ch2. URL https://onlinelibrary.wiley.com/doi/10.1002/9780470316801.ch2.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kornblith19a.html.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2757–2791, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.138. URL https://aclanthology.org/2025.emnlp-main.138/.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL https://aclanthology.org/2022.acl-long.229/.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers, 2021. URL https://arxiv.org/abs/2106.04554.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding, 2017. URL https://arxiv.org/abs/1703.03130.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 3622–3628. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/501. URL https://doi.org/10.24963/ijcai.2020/501. Main track.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020. URL https://arxiv.org/abs/2006.15055.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/.

NVIDIA Corporation. Nvidia tesla v100 gpu accelerator — data sheet. Technical report, NVIDIA, March 2018. URL https://images.nvidia.com/content/technologies/volta/pdf/tesla-volta-v100-datasheet-letter-fnl-web.pdf.

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

Team OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann (eds.), *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pp. 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

Shivam Patel, Neharika Jali, Ankur Mallick, and Gauri Joshi. Proxrouter: Proximity-weighted llm query routing for improved robustness to outliers, 2025. URL https://arxiv.org/abs/2510.09852.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Robert J. Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, 2(1):39–48, 1974.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16857–16867. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/c3a690be93aa602ee2dc0ccab5b7b67e-Paper.pdf.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), December 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

Nilesh Tripuraneni, Michael I. Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Yun-Yun Tsai, Chuan Guo, Junfeng Yang, and Laurens van der Maaten. Rofl: Robust fingerprinting of language models, 2025. URL https://arxiv.org/abs/2505.12682.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 5776–5788. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Zhaomin Wu, Haodong Zhao, Ziyang Wang, Jizhou Guo, Qian Wang, and Bingsheng He. Llm dna: Tracing model evolution via functional representations, 2025. URL https://arxiv.org/abs/2509.24496.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *CoRR*, abs/2402.13116, 2024. URL https://doi.org/10.48550/arXiv.2402.13116.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1480–1489, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1174. URL https://aclanthology.org/N16-1174/.

Louie Hong Yao, Nicholas Jarvis, Tiffany Zhan, Saptarshi Ghosh, Linfeng Liu, and Tianyu Jiang. Je-irt: A geometric lens on llm abilities through joint embedding item response theory, 2025. URL https://arxiv.org/abs/2509.22888.

Boyi Zeng, Lizheng Wang, Yuncong Hu, Yi Xu, Chenghu Zhou, Xinbing Wang, Yu Yu, and Zhouhan Lin. Huref: HUman-REadable fingerprint for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=RlZgnEZsOH.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, Jia Xu, Lei Bai, Wanli Ouyang, and Shuyue Hu. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants, 2025. URL https://arxiv.org/abs/2505.19797.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, 2025.

A Language Models and Query Datasets

We use the model-query evaluations presented in Zhuang et al. (2025). The queries are sampled from ten public benchmarks, representing a wide variety of real-world prompts and tasks: **MathQA** Amini et al. (2019), **LogiQA** Liu et al. (2020), **MedMCQA** Pal et al. (2022), **PIQA** Bisk et al. (2020), **TruthfulQA** Lin et al. (2022), **MMLU** Hendrycks et al. (2021), **GSM8K** Cobbe et al. (2021), **GPQA** Rein et al. (2024), **ASDiv** Miao et al. (2020), **SocialQA** Sap et al. (2019). Semantic query encodings generated by sentence encoders are presented in Figure 10 for four choices of sentence encoders, colored by dataset.

The 112 language models used are presented in Table 2 with their HuggingFace identifiers Wolf et al. (2020). For new model onboarding analysis as presented in Section 4.2 and Figure 3, we consider the following sixteen models as held-out from training (seen only during test time): `FelixChao/llama2-13b-math1.2`, `MaziyarPanahi/WizardLM-Math-70B-v0.1`, `01-ai/Yi-6B-200K`, `WizardLM/WizardLM-70B-V1.0`, `bigscience/bloom-7b1`, `sail/Sailor-7B`, `codellama/CodeLlama-13b-Instruct-hf`, `Writer/palmyra-med-20b`, `Qwen/Qwen1.5-0.5B-Chat`, `databricks/dolly-v2-12b`, `nomic-ai/gpt4all-13b-snoozy`, `stabilityai/stablelm-tuned-alpha-7b`, `AdaptLLM/medicine-chat`, `AdaptLLM/medicine-LLM`, `EleutherAI/pythia-12b`, `Q-bert/Optimus-7B`.

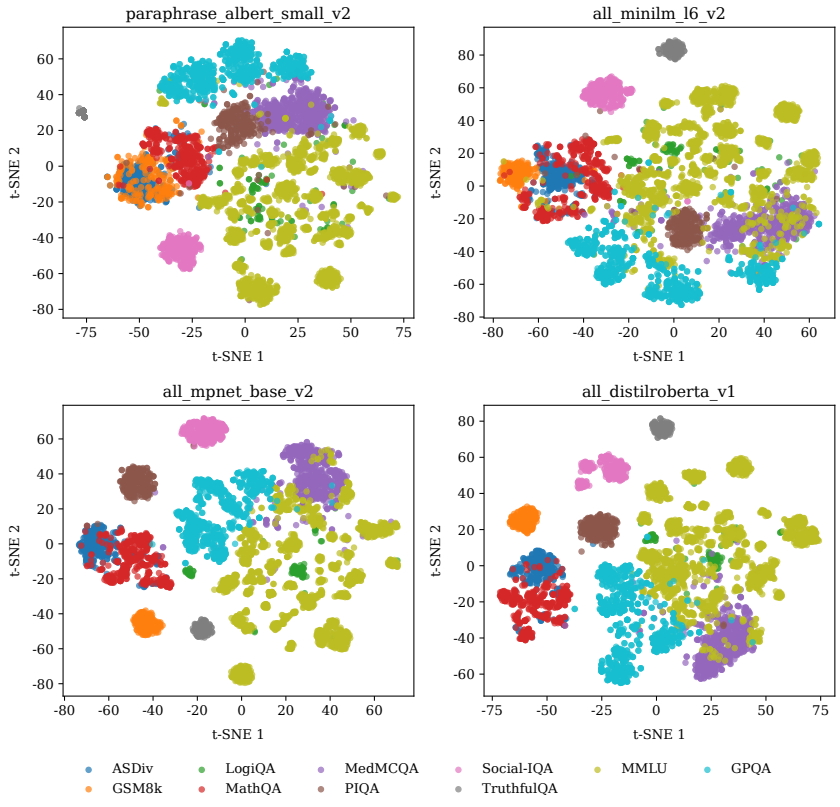


Figure 10: Query encodings visualized using t-SNE van der Maaten & Hinton (2008), colored by dataset. Queries from the same dataset represent similar tasks, and their sentence encodings are placed close by in the encoding space. Plots presented for four different sentence encoders `all_mpnet_base_v2` Song et al. (2020), `paraphrase_albert_small_v2` Lan et al. (2020), `all_minilm_l6_v2` Wang et al. (2020), and `all_distilroberta_v1` Sanh et al. (2020).

Table 2: Language models utilized in our experiments, as presented in Zhuang et al. (2025). All 112 language models are denoted by their HuggingFace identifiers Wolf et al. (2020).

Qwen/Qwen1.5-7B-Chat	ConvexAI/Luminex-34B-v0.1
lmsys/vicuna-13b-v1.5	deepseek-ai/deepseek-math-7b-instruct
TigerResearch/tigerbot-13b-base	ConvexAI/Luminex-34B-v0.2
berkeley-nest/Starling-LM-7B-alpha	EleutherAI/llemma_7b
CultriX/NeuralTrix-bf16	SciPhi/SciPhi-Mistral-7B-32k
TheBloke/tulu-30B-fp16	lmsys/vicuna-33b-v1.3
scb10x/typhoon-7b	mlabonne/AlphaMonarch-7B
mistralai/Mistral-7B-Instruct-v0.1	01-ai/Yi-34B-Chat
meta-llama/Llama-2-13b-chat-hf	eren23/ogno-monarch-jaskier-merge-7b-OH-PREF-DPO
ibivibiv/alpaca-dragon-72b-v1	golaxy/gowizardlm
codellama/CodeLlama-34b-Instruct-hf	OpenBuddy/openbuddy-codellama2-34b-v11.1-bf16
deepseek-ai/deepseek-coder-1.3b-base	Neko-Institute-of-Science/pygmalion-7b
cognitivecomputations/yayi2-30b-llama	meta-llama/LlamaGuard-7b
NousResearch/Nous-Hermes-13b	tiiuae/falcon-40b-instruct
meta-llama/Llama-2-7b-chat-hf	mosaicml/mpt-7b-chat
Qwen/Qwen1.5-32B-Chat	NousResearch/Nous-Hermes-2-Yi-34B
deepseek-ai/deepseek-coder-6.7b-instruct	google/gemma-7b-it
EleutherAI/llemma_34b	zhengr/MixTAO-7Bx2-MoE-v8.1
yam-peleg/Experiment26-7B	meta-llama/Meta-Llama-3-8B
mosaicml/mpt-30b-instruct	fblgit/UNA-SimpleSmaug-34b-v1beta
FelixChao/vicuna-7B-physics	TheBloke/koala-13B-HF
meta-llama/Meta-Llama-3-70B	Plaban81/Moe-4x7b-math-reason-code
meta-math/MetaMath-Mistral-7B	BioMistral/BioMistral-7B
FelixChao/Scorpio-7B	SciPhi/SciPhi-Self-RAG-Mistral-7B-32k
microsoft/phi-2	CausalLM/34b-beta
meta-llama/Meta-Llama-3-70B-Instruct	meta-math/MetaMath-Llemma-7B
lmsys/vicuna-7b-v1.5-16k	cloudyu/Mixtral_11Bx2_MoE_19B
Qwen/Qwen1.5-4B-Chat	FelixChao/vicuna-7B-chemical
HuggingFaceH4/zephyr-7b-beta	OpenAssistant/oasst-sft-4-pythia-12b-epoch-3.5
BioMistral/BioMistral-7B-DARE	Biomimicry-AI/ANIMA-Nectar-v2
microsoft/phi-1_5	meta-llama/Meta-Llama-Guard-2-8B
rishiraj/CatPPT-base	kyujinpy/Sakura-SOLRCA-Math-Instruct-DPO-v1
meta-llama/Meta-Llama-3-8B-Instruct	google/gemma-2b-it
upstage/SOLAR-10.7B-Instruct-v1.0	CorticalStack/pastiche-crown-clown-7b-dare-dpo
01-ai/Yi-6B	codefuse-ai/CodeFuse-DeepSeek-33B
abhishek/zephyr-beta-math	bardsai/jaskier-7b-dpo-v5.6
allenai/tulu-2-dpo-70b	Harshvir/Llama-2-7B-physics
lmsys/vicuna-13b-v1.5-16k	shleeeee/mistral-ko-tech-science-v1
JaeyeonKang/CCK_Asura_v1	codellama/CodeLlama-7b-hf
Nexusflow/Starling-LM-7B-beta	microsoft/Orca-2-13b
Neko-Institute-of-Science/metharme-7b	bigcode/octocoder
PharMolix/BioMedGPT-LM-7B	SUSTech/SUS-Chat-34B
kevin009/llamaRAGdrama	meta-llama/Llama-2-70b-chat-hf
TheBloke/CodeLlama-70B-Instruct-AWQ	openchat/openchat_3.5
dfurman/HermesBagel-34B-v0.1	project-baize/baize-v2-13b
augmxnt/shisa-base-7b-v1	lmsys/vicuna-7b-v1.5
Intel/neural-chat-7b-v3-3	AdaptLLM/medicine-LLM-13B
openchat/openchat-3.5-0106	deepseek-ai/deepseek-llm-67b-chat
FelixChao/llama2-13b-math1.2	MaziyarPanahi/WizardLM-Math-70B-v0.1
01-ai/Yi-6B-200K	WizardLM/WizardLM-70B-V1.0
bigscience/bloom-7b1	sail/Sailor-7B
codellama/CodeLlama-13b-Instruct-hf	Writer/palmyra-med-20b
Qwen/Qwen1.5-0.5B-Chat	databricks/dolly-v2-12b
nomic-ai/gpt4all-13b-snoozy	stabilityai/stablelm-tuned-alpha-7b
AdaptLLM/medicine-chat	AdaptLLM/medicine-LLM
EleutherAI/pythia-12b	Q-bert/Optimus-7B

B Extended Numerical Evaluation

B.1 Evaluation of Starmap on Continuous Scoring Tasks

In the main empirical analysis presented in Section 4, we utilize datasets which have binary correctness labels (as the tasks are verifiable and responses are compared with ground truths). In Table 3, we conduct empirical examination over tasks that are evaluated using LLM-as-a-judge Li et al. (2025), describing the generalizability of STARMAP beyond binary scoring approaches.

Table 3: Overall routing accuracy (%) and correctness accuracy (score prediction) across approaches and training-set sizes (number of evaluation queries) for continuous scoring datasets Chinese_Idioms, Consensus_Summary, Grade-School-Math (stepwise marking), Mtbench-Reference, Mtbench, Mtbench-math.

Approach (# Query Evaluations)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
STARMAP (Ours)	72.28	72.95	73.29	71.99	72.65	72.63
EMBEDLLM	67.48	72.34	70.93	70.54	71.35	72.99
IRT-NET	72.62	72.87	71.69	72.48	72.05	73.01
KNN	68.07	70.09	70.85	70.96	71.68	72.76

B.2 Additional Comparison with Baselines

We present the numerical results on routing accuracy and correctness prediction (Table 4) of STARMAP, compared against other baselines: EmbedLLM Zhuang et al. (2025), and IRT-Net Chen et al. (2025). We additionally present correctness prediction and routing accuracy of a popular nonparametric router design which uses kNN based aggregation to estimate model performance and cost on test queries Hu et al. (2024); Jitkrittum et al. (2025); Patel et al. (2025); Zhang et al. (2025).

We observe that across all model embedding based approaches, STARMAP exhibits the highest overall routing accuracy across all choices of number of training samples (number of queries each model is evaluated on). This is succinctly described in Figure 1 as well. For correctness prediction accuracy, we find that all approaches perform similarly in terms of overall correctness prediction accuracy.

Table 4: Overall routing and correctness prediction accuracy (%) across approaches and training-set sizes.

Approach (# Train Samples)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
STARMAP (OURS)	61.90	62.97	64.70	68.31	68.33	70.03
EMBEDLLM	58.80	59.47	59.60	67.33	68.12	69.47
IRT-NET	59.57	60.17	63.37	67.38	69.07	70.12
KNN	60.50	61.37	62.90	60.60	67.90	69.27

We compare STARMAP with LLM-DNA Wu et al. (2025) in Table 5. LLM-DNA utilizes response embeddings to generate model embeddings by Gaussian downprojection, and train an MLP model on top of the embeddings to perform correctness prediction. The EmbedLLM Zhuang et al. (2025) dataset used in our main analysis only has response scores and not text responses themselves. Not all of the language models analyzed in EmbedLLM are available on the HuggingFace repository (due to being taken down) at the time of this writing, so we compare STARMAP with LLM-DNA on ProxRouter Patel et al. (2025) data. We observe that STARMAP outperforms LLM-DNA on routing accuracy and correctness prediction over all number of queries used for generating the model embeddings.

Table 5: Overall routing and correctness prediction accuracy (%) comparison with LLMDNA Wu et al. (2025) and our approach on ProxRouter Patel et al. (2025) dataset.

Approach (# Train Samples)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
STARMAP (OURS)	80.79	82.78	82.11	63.83	67.50	67.19
LLMDNA	70.53	63.58	63.91	61.45	60.38	60.48

C ABLATION ON CHOICE OF SENTENCE ENCODER FOR QUERY REPRESENTATION

In this section, we present correctness prediction accuracy and routing accuracy of STARMAP through different choices of sentence encoders for generating query representations. In our main analysis, we select a popular sentence encoder `all_mpnet_base_v2` Song et al. (2020) which produces query representations $\in \mathbb{R}^{768}$.

Table 6: Overall routing and correctness prediction accuracy (%) across query sentence encoder choices.

Sentence Encoder (# Train Samples)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
`all_mpnet_base_v2`	61.90	62.97	64.70	68.31	68.33	70.03
`paraphrase_albert_small_v2`	59.00	61.40	60.97	59.42	66.61	67.69
`all_minilm_l6_v2`	61.60	63.20	62.27	66.12	68.14	69.28
`all_distilroberta_v1`	61.83	62.60	63.33	67.50	68.17	70.34

Here, we present results for `paraphrase_albert_small_v2` Lan et al. (2020) ($\in \mathbb{R}^{768}$), `all_minilm_l6_v2` Wang et al. (2020) ($\in \mathbb{R}^{384}$), and `all_distilroberta_v1` Sanh et al. (2020) ($\in \mathbb{R}^{768}$). Our model architecture is uniform across all sentence encoders (except for different input dimensions across choices of sentence encoders). We present routing accuracy and correctness prediction accuracy in Table 6. We observe that overall routing accuracy and correctness prediction accuracy is comparable across sentence encoders.

D ABLATION ON HYPERPARAMETER CHOICES

Here, we present ablations on the routing accuracy and correctness prediction performance of STARMAP (trained on 256 query evaluations) with varying number of model embedding dimension (Table 7), varying the number of layers in the encoder (Table 8), varying the number of latent bottleneck attention vectors (Table 9), and varying the number of heads in each attention layer (Table 10)

Table 7: Overall routing and correctness prediction accuracy (%) for STARMAP with varying the model embedding dimension.

Embedding Dim	Routing Accuracy (%)	Corr. Pred Acc (%)
16	60.97	67.94
32	62.13	67.75
64	61.77	67.82
128 (default)	61.90	68.31
256	61.87	68.42
512	62.10	67.93

Table 8: Overall routing and correctness prediction accuracy (%) for STARMAP with varying number of layers in the encoder.

Number of Layers	Routing Accuracy (%)	Corr. Pred Acc (%)
1 Layer	62.06	68.15
2 Layers (default)	61.90	68.31
3 Layers	61.96	68.22
4 Layers	62.60	67.89

Table 9: Overall routing and correctness prediction accuracy (%) for STARMAP with varying number of latent bottleneck attention vectors in the encoder.

Number of Vectors	Routing Accuracy (%)	Corr. Pred Acc (%)
16	62.46	67.96
64 (default)	61.90	68.31
128	62.36	67.76
256	58.86	68.11

E ADDITIONAL UTILITY OF MODEL EMBEDDINGS

E.1 RETRIEVING MODELS THAT MATCH DESIRED TASK PROFILES VIA HYPOTHETICAL EMBEDDINGS.

We next test whether embeddings preserve task-level performance structure rather than overfitting to query identity. For each model $m \in \mathcal{M}$, we compute its empirical task-wise accuracy profile across a set of tasks. We then construct a *hypothetical* evaluation set by sampling queries and assigning synthetic correctness labels so that the resulting task-wise accuracies match (in expectation) the target profile of m. Feeding this synthetic set through the encoder produces a hypothetical embedding $\tilde{z}_m$, which we then compare against the library of real model embeddings $\{z_m\}_{m\in\mathcal{M}}$ via nearest-neighbor search.

We report recall@k: the fraction of models for which the true model embedding z_m appears among the top-k nearest real embeddings to $\tilde{z}_m$. Averaged over all models, recall@10 is almost 97% for evaluation sets of size 8192 queries, depicting that model embeddings capture high level task performance (Figure 11). These results indicate that the embedding generator maps evaluation sets to representations that reflect higher-level behavioral profiles: even when correctness labels are randomized at the query level subject to matching task-level rates, the resulting embedding is often close to the intended model.

E.2 RESILIENT FALLBACK ROUTING UNDER MODEL UNAVAILABILITY.

In deployed query routing systems, the language model selected by a router may become temporarily unreachable (e.g., due to overload, failures, or policy constraints). When routing decisions are produced by a black-box router and per-query correctness probabilities / suitability scores for models are not readily available, a natural fallback is to substitute the router selected model with a nearby model in embedding space. To study the validity of the fallback routing approach, for each test query x we take the routed model $\widehat{m}(x)$ and simulate its unavailability, then generate a response using its kth nearest neighbor by embedding distance (closest, second-closest, etc.). Figure 12 reports routing accuracy versus neighbor rank: the closest-neighbor fallback retains up to $\mathbf{85}\%$ of the original routing accuracy, with performance decreasing smoothly as k increases. This supports embedding proximity as a practical mechanism for resilient fallback under model outages.

Table 10: Overall routing and correctness prediction accuracy (%) for STARMAP with varying number of Attention heads in a Multihead Attention layer.

Number of Heads	Routing Accuracy (%)	Corr. Pred Acc (%)
1 Head	61.83	68.20
2 Heads	57.90	59.43
4 Heads (Default)	61.90	68.31
8 Heads	43.46	59.42

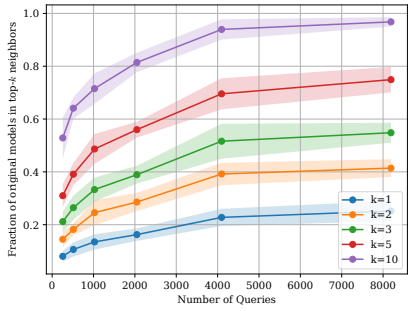


Figure 11: We obtain model embeddings based on random assignment of correctness labels to queries to match dataset-wise accuracy scores of actual LLMs in the pool. Then we find the proportion of original model embeddings that are among the $(1, 2, 3, 5, 10)$ nearest neighbors of the new generated embedding. A high fraction of original embeddings present in top-k neighbors of the new generated embeddings denotes that STARMAP captures task level behavior in model embedding geometry.

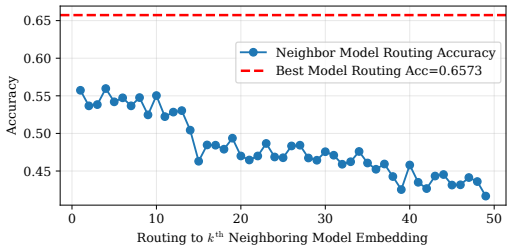


Figure 12: **Resilient Routing to Fallback Models.** Selected models for query routing might not always be available due to congestion, node failure etc. Routing to next closest model embedding provides a fallback mechanism, while maintaining up to 85% of overall routing performance. Performance gracefully decays with chosen rank of neighboring model to route queries to.

E.3 CONVERGENCE RELATIVE TO INTER-MODEL SEPARATION, AND IMPLICATIONS TO SECURITY FINGERPRINTING.

To quantify and contextualize the model embedding drift w.r.t. varying the evaluation set overlap and size, we plot the distance to the originally generated reference embedding $\|z_m - z_m^{\text{ref}}\|$ and compare them with embedding distances of reference embedding of chosen model to reference embeddings of other distinct models. Figure 13 shows that the new model embeddings generated are close to the reference embeddings (as compared to distances to other reference model embeddings), and the distance/dissimilarity decays as the overlap fraction (or the number of evaluation queries) increases. A similar trend is also observed for subsampling a smaller number of queries in the evaluation set for generating model embeddings, as presented in Figure 14.

Relative robustness of generated model embeddings to the choice of queries demonstrates possible applications in security fingerprinting of language models, where very close embeddings between two models can indicate the same language model hidden behind different APIs or endpoints.

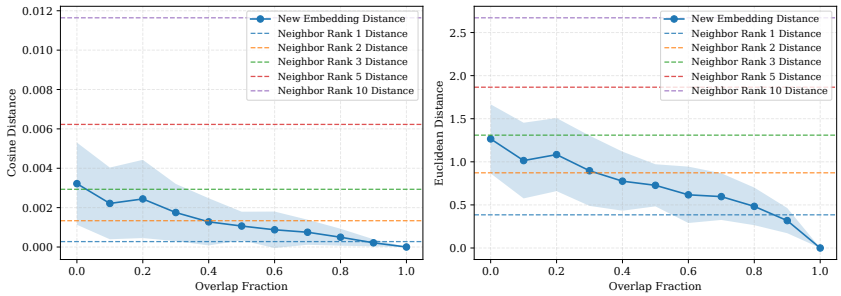


Figure 13: Distance between new generated model embedding to reference embedding (for CodeLlama-13b-hf model) w.r.t. changing the overlap fraction of queries in evaluation set used for generating model embeddings. Horizontal dashed lines denote distance of chosen model's reference embedding to other models' embeddings.

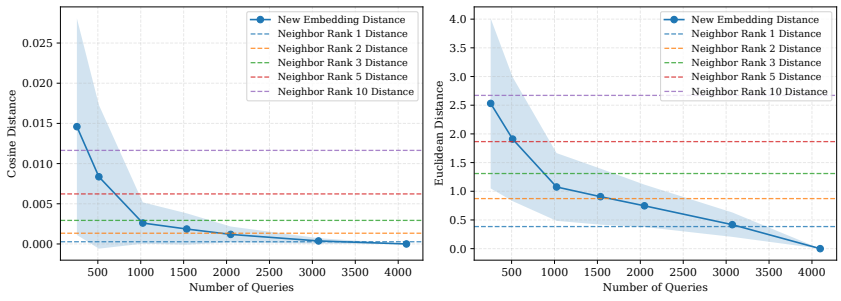


Figure 14: Distance between new generated model embedding to reference embedding (for CodeLlama-13b-hf model) w.r.t. changing the number of queries in evaluation set used for generating model embeddings. Horizontal dashed lines denote distance of chosen model's reference embedding to other models' embeddings.

E.4 Model portfolio selection using embedding geometry

E.4.1 Setup and portfolio routing accuracy

Let $\mathcal{M} = \{1, \dots, M\}$ denote the full model pool and let $z_m \in \mathbb{R}^d$ be the learned embedding for model m. Given a query x, the router assigns a correctness probability $\widehat{p}^{(m)}(x)$ to each model m. For a selected portfolio $\mathcal{M}_{\text{sel}} \subseteq \mathcal{M}$, routing chooses

$$\widehat{m}_{\text{sel}}(x) \;=\; \arg\max_{m \in \mathcal{M}_{\text{sel}}} \widehat{p}^{(m)}(x). \tag{5}$$

Let $y^{(m)}(x) \in \{0, 1\}$ indicate whether model m answers x correctly (used only for evaluation). We define *portfolio routing accuracy* as

$$\text{Acc}(\mathcal{M}_{\text{sel}}) \;=\; \mathbb{E}_x\big[y^{(\widehat{m}_{\text{sel}})}(x)\big], \tag{6}$$

which we estimate on held-out data as routing accuracy after restricting to the models in $\mathcal{M}_{\text{sel}}$. Our goal is to select $\mathcal{M}_{\text{sel}}$ using *only* embedding geometry $\{z_m\}$ under two different deployment constraints:

1. a *count* constraint $|\mathcal{M}_{\text{sel}}| \le k$, with $k \in \{1, 2, 3, \cdots\}$
2. a *parameter budget* constraint $\sum_{m \in \mathcal{M}_{\text{sel}}} c_m \le \mathcal{C}$, where $c_m > 0$ denotes total number of parameters in model m.

Similarity between model embeddings through Kernels. Let $K(z_i, z_j) \in [0, 1]$ be a similarity kernel in the model embedding space (e.g., an RBF kernel), where larger values indicate close model embeddings and hence similar model behaviors. Concretely, we define

$$K(z_i, z_j) \;=\; \exp\Big(-\tfrac{d_z(z_i, z_j)^2}{\sigma^2}\Big), \tag{7}$$

with $d_z(\cdot, \cdot)$ being a distance/dissimilarity measure (e.g., cosine distance) and bandwidth $\sigma > 0$ (which we set as the median pairwise distance between model embeddings).

E.4.2 Model Count Constrained Selection

To choose a subset of models such that collective abilities of the pool is maintained, we select model embeddings spread across the embedding space via the kernelized k-center objective Gonzalez (1985):

$$\max_{\mathcal{M}_{\text{sel}} \subseteq \mathcal{M}:\ |\mathcal{M}_{\text{sel}}| \le k} \;\min_{i \in \mathcal{M}} \;\max_{s \in \mathcal{M}_{\text{sel}}} K(z_i, z_s), \tag{8}$$

which seeks to choose k model embeddings such that every model in the full pool has *high similarity* to at least one chosen model in the subset (i.e., the worst-covered model is as well-covered as possible).

We additionally study the kernelized k-medoids objective Kaufman & Rousseeuw (1990) which maximizes the sum similarity of all model embeddings to the most similar among the chosen subset of models closest :

$$\max_{\mathcal{M}_{\text{sel}} \subseteq \mathcal{M}:\ |\mathcal{M}_{\text{sel}}| \le k} \;\sum_{i \in \mathcal{M}} \;\max_{s \in \mathcal{M}_{\text{sel}}} K(z_i, z_s), \tag{9}$$

which selects k embeddings so that the *total coverage* (best similarity per model, summed over the pool) is maximized.

Both aforementioned approaches depend only on model embedding geometry and do not utilize per-query correctness.

`k-center` (farthest-first in similarity space) Gonzalez (1985). We apply a similarity-space analogue of farthest-first sampling. We initialize $\mathcal{M}_{\text{sel}} = \{m_0\}$ using the *global kernel medoid*

$$m_0 \;=\; \arg\max_{m \in \mathcal{M}} \;\frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} K(z_i, z_m), \tag{10}$$

then iteratively add the *least-covered* model (the one whose best similarity to the current set is smallest):

$$m^\star \;=\; \arg\min_{m\in\mathcal{M}\setminus\mathcal{M}_{\text{sel}}} \; \max_{s\in\mathcal{M}_{\text{sel}}} K(z_m, z_s), \qquad \mathcal{M}_{\text{sel}} \leftarrow \mathcal{M}_{\text{sel}} \cup \{m^\star\}. \tag{11}$$

Intuitively, this repeatedly adds a representative for the region of embedding space that is poorly covered at each step (lowest similarity to models in the selected set $\mathcal{M}_{\text{sel}}$).

`k-medoids` Kaufman & Rousseeuw (1990). Initializing with the `k-center` solution, we utilize the PAM (Partitioning Around Medoids) refinement algorithm which swaps model embeddings under sum-similarity objective and replaces embeddings which result in improvement of the objective.

Define the sum-similarity objective as

$$J(\mathcal{M}_{\text{sel}}) \;=\; \sum_{i\in\mathcal{M}} \max_{s\in\mathcal{M}_{\text{sel}}} K(z_i, z_s), \tag{12}$$

so that maximizing J is equivalent to maximizing the total similarity coverage in eq. (9). PAM searches for swaps $s \in \mathcal{M}_{\text{sel}}$ and $h \notin \mathcal{M}_{\text{sel}}$ that improve the objective value:

$$\mathcal{M}'_{\text{sel}} \;=\; (\mathcal{M}_{\text{sel}} \setminus \{s\}) \cup \{h\}, \qquad \text{accept if } J(\mathcal{M}'_{\text{sel}}) > J(\mathcal{M}_{\text{sel}}), \tag{13}$$

and stops when no improving swap exists (or after a fixed number of iterations).

Random baseline. As a baseline, we sample a subset of models $\mathcal{M}_{\text{sel}} \subseteq \mathcal{M}$ of size k uniformly at random, and report mean $\pm$ standard deviation in routing accuracy across trials.

Evaluation. For each k, we select $\mathcal{M}_{\text{sel}}$ using only $\{z_m\}$ (via the kernel $K(\cdot,\cdot)$) and evaluate $\mathrm{Acc}(\mathcal{M}_{\text{sel}})$ by restricting routing accuracy to the chosen subset of models.

Observations. For the count constraint model portfolio selection problem, we find that simple approaches like `k-medoids` and `k-center` (Figure 9) achieve full model pool (comprising 112 models) routing accuracy with just $15 - 20$ models in the selected subset, signifying that selecting a subset of models by maximizing coverage over the model embedding space leads to capturing the collective abilities of the entire model pool. Such efficient selection of capable model subsets/portfolios points towards an efficient management of a large pool of language models, and eliminating redundancy in model abilities for simplifying deployment and improving hardware utilization. We reiterate that the presented approaches do not require any query-correctness prediction over models on representative queries, and rather the model subset choice is made purely based on model embedding space.

E.4.3 Parameter Budget Constraint Selection

We utilize the sum-similarity objective as presented above in eq. (12) and solve the parameter-budget-constrained maximization for respecting total parameter budget for model subset choice

$$\max_{\mathcal{M}_{\text{sel}}\subseteq\mathcal{M}} \; J(\mathcal{M}_{\text{sel}}) \quad \text{s.t.} \quad \sum_{m\in\mathcal{M}_{\text{sel}}} c_m \le \mathcal{C}. \tag{14}$$

Greedy selection (marginal gain per parameter). For simplicity, we study the greedy solution to the formulation in eq. (14). Let $b_i = \max_{s\in\mathcal{M}_{\text{sel}}} K(z_i, z_s)$ denote most similar selected model in $\mathcal{M}_{\text{sel}}$ for model i. For a candidate $m \notin \mathcal{M}_{\text{sel}}$, the marginal gain is

$$\Delta(m \mid \mathcal{M}_{\text{sel}}) \;=\; J(\mathcal{M}_{\text{sel}} \cup \{m\}) - J(\mathcal{M}_{\text{sel}}) \;=\; \sum_{i\in\mathcal{M}} \max\big\{0,\; K(z_i, z_m) - b_i\big\}. \tag{15}$$

Under the remaining budget, we add the model with the largest gain per parameter:

$$m^\star \;=\; \arg\max_{\substack{m\notin\mathcal{M}_{\text{sel}},\\ c_m \le\, (\mathcal{C}-\sum_{s\in\mathcal{M}_{\text{sel}}} c_s)}} \frac{\Delta(m \mid \mathcal{M}_{\text{sel}})}{c_m}, \qquad \mathcal{M}_{\text{sel}} \leftarrow \mathcal{M}_{\text{sel}} \cup \{m^\star\}, \tag{16}$$

and stop when no feasible model remains.

Evaluation. For each parameter count constraint $\mathcal{C}$, we select $\mathcal{M}_{\text{sel}}^{\mathcal{C}}$ using eq. (16) and evaluate $\text{Acc}(\mathcal{M}_{\text{sel}}^{\mathcal{C}})$ by restricting evaluation to models present in $\mathcal{M}_{\text{sel}}^{\mathcal{C}}$. We also compare to a *random feasible* baseline that samples subsets satisfying $\sum_{m \in S} c_m \leq \mathcal{C}$ and reports mean $\pm$ standard deviation across trials.

Observations. For the parameter constrained choice of models (Figure 9), we note that the described greedy algorithm which maximizes coverage over the space of model embeddings consistently outperforms random choice of models, attaining very close to full model pool routing accuracy at $\approx$150B parameters (full model pool has 112 models, with 1930B parameters in total). For practical deployment scenarios with limited hardware, clever choice of subset of models achieves almost full model pool's routing accuracy by eliminating duplicated model capabilities (in total just 8% parameter budget across all models in the pool).

F Regenerating Model Embeddings for Backpropagation-based Approaches

In this section, we describe the shortcomings of learnable embeddings based approaches. For the sake of brevity, we consider EmbedLLM Zhuang et al. (2025) and examine the geometry and capability representation of model embeddings learned by original training vs retraining/regenerating embeddings with the same data. Concretely, consider $\{e_\theta^{(m)}\}_{m \in \mathcal{M}}$ as the trainable embeddings for each model in the pool, and let G_ψ denote the correctness predictor which is of the form $G_\psi(e_\theta^{(m)}, x) = \text{MLP}_\psi(e_\theta^{(m)} \odot W_\psi x)$, where x is the query encoding generated through a sentence encoder, W_ψ is the trained query downprojection matrix, MLP_ψ is a multilayer perceptron network, and $\odot$ denotes elementwise product. We perform training over all model-query evaluation pairs, to minimize the binary cross entropy loss between predictions and observed correctness values from evaluations:

$$\min_{\theta, \psi} \quad \mathbb{E}_m \, \mathbb{E}_x \, \Big[\text{BCE}\big(G_\psi(e_\theta^{(m)}, x), \; y^{(m)}(x)\big)\Big] \tag{17}$$

This results in learned model embeddings $\{e_\theta^{(m)}\}_{m \in \mathcal{M}}$, and correctness predictor G_ψ. To evaluate the stability of the generated model embeddings, we freeze the correctness predictor parameters G_ψ, and then retrain the model embeddings to obtain $\{e_{\theta_{\text{re}}}^{(m)}\}_{m \in \mathcal{M}}$ with the same model query evaluation data. Mathematically, we perform:

$$\min_{\theta_{\text{re}}} \quad \mathbb{E}_m \, \mathbb{E}_x \, \Big[\text{BCE}\big(G_\psi(e_{\theta_{\text{re}}}^{(m)}, x), \; y^{(m)}(x)\big)\Big] \tag{18}$$

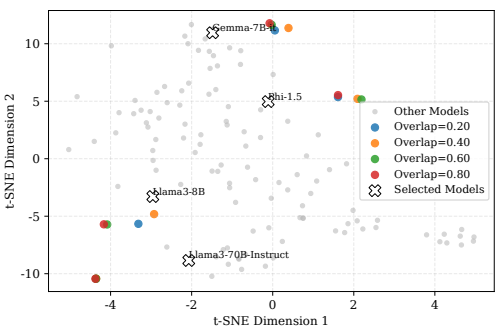
This represents a practical scenario where unseen models are added to the pool, and embeddings need to be generated for them without affecting correctness predictor G_ψ and already learned embeddings $\{e_\theta^{(m)}\}_{m \in \mathcal{M}}$ for models in the pool. For the sake of analysis, we consider new added models as clones of existing models with identical query evaluation data, and hence compare their newly generated embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m \in \mathcal{M}}$ with originally trained embeddings $\{e_\theta^{(m)}\}_{m \in \mathcal{M}}$ for similarity.

Because the training objective in eq. (18) differs from that in eq. (17), we anticipate small changes in the regenerated model embeddings, even though the embeddings are learned on the same query evaluation data. However, they should remain close to the original embeddings to preserve the practically desirable properties described in Section 1. If the regenerated embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m \in \mathcal{M}}$ are very different as compared to original model embeddings $\{e_\theta^{(m)}\}_{m \in \mathcal{M}}$, we cannot utilize informative embedding geometry. In such scenarios, we cannot mutually compare model embeddings across models to identify similarities or differences in capabilities, making most of the analyses and practical applications presented in Section 4 infeasible.

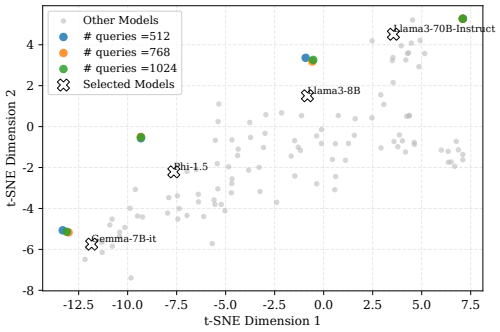
We examine EmbedLLM Zhuang et al. (2025) through a similar approach as described in Section 4.3, where we regenerate model embeddings through new query evaluation sets to test sensitivity to *which* queries are used for generating embeddings and *how many* queries are used to generate embeddings. Figure 15 denotes geometric representation of original and regenerated embeddings (cf. Section 4.3), highlighting that regenerated embeddings are not in close proximity of the original embeddings. This

is also presented numerically in Figure 16, where average cosine distance between respective original and regenerated embeddings is high, even though regenerated embeddings are trained on the same evaluation data as original embeddings.

Discrepancy between original and regenerated embeddings also affects correctness prediction over queries. Figure 17 denotes average correctness prediction disagreement (over a common set of test queries) by using original versus regenerated embeddings with the same correctness predictor G_ψ. Again, regenerated embeddings are trained on the original evaluation set, but still display up to 8.1% disagreement rate on predicting correctness values for test queries.



(a) Regenerating model embeddings using query evaluations with varying overlap to original data.



(b) Regenerating model embeddings using queries sampled as a subset of original evaluation data.

Figure 15: Geometrically comparing original $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ and regenerated embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m\in\mathcal{M}}$ for varying overlap fraction with original evaluation set, and subsampling fewer queries from original evaluation set. Regenerated embeddings are far away from original model embeddings in both cases, denoting weakly informative embedding geometry.

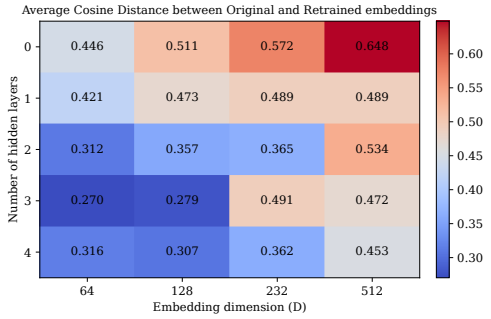


Figure 16: Average cosine distance between initially trained model embeddings $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ and regenerated model embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m\in\mathcal{M}}$. We present results across number of hidden layers in MLP_ψ (with $d_{\text{hidden}} = 64$), along with chosen embedding dimension. Regenerated model embeddings exhibit high cosine distance to original trained embeddings even if retrained on same query evaluation data, rendering the model embedding geometry brittle for practical applications.

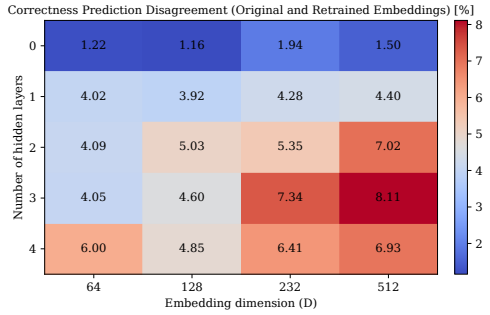
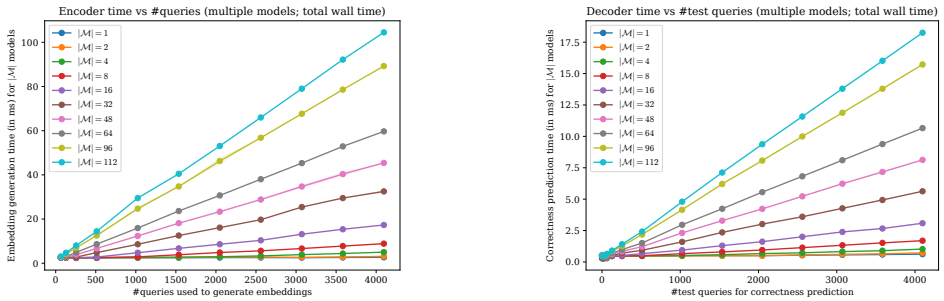


Figure 17: Query Correctness Disagreement between initially trained model embeddings $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ and regenerated model embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m\in\mathcal{M}}$. We present the absolute percentage disagreement on correctness prediction averaged over all models for a common set of queries, across number of hidden layers in MLP_ψ (with $d_{\text{hidden}} = 64$), along with chosen embedding dimension. High correctness disagreement between original and regenerated embeddings denotes non-identical capability representation, even if trained on the same query evaluation data.

G TIME COMPLEXITY OF STARMAP FOR GENERATING EMBEDDINGS AND CORRECTNESS PREDICTION

In this section, we examine the time complexity of STARMAP for generating embeddings using query evaluations, and for correctness prediction on unseen queries. We utilize a single NVIDIA Tesla V100 (32GB) GPU NVIDIA Corporation (2018) for our experiments.



(a) Model Embedding Generation (via F_θ)

(b) Unseen Query correctness prediction (via G_ψ)

Figure 18: **Time Complexity of STARMAP (in ms).** ***(a)*** We consider embedding generation for different number of models ($|\mathcal{M}|$), using varying query evaluation sets. Linear time complexity (in terms of # evaluation queries used) of our Latent Bottleneck Attention blocks (eq. (2) and Section 2.1) is evident from above. ***(b)*** The time complexity of our correctness predictor is of the order of ≈20ms, even for calculating probability of correctness over the entire model pool (with 112 models) for a batch of 4096 unseen queries. This denotes the efficiency of our correctness predictor for practical tasks such as routing, where the additional correctness prediction overhead is orders of magnitude smaller than the actual response generation time (usually in seconds (Kwon et al., 2023)).

Figure 18a denotes the wall clock time required to generate model embeddings (for $|\mathcal{M}|$ models in parallel) using query evaluation sets of varying sizes. We observe that even for parallel embedding generation for the full model pool (112 models) on 4096 queries, required time is ≈100ms. This indicates powerful yet efficient design of STARMAP for generating model embeddings from observed query evaluations. Additionally, Figure 18b denotes the correctness prediction time for varying number of unseen queries using the decoder G_ψ. Calculating correctness probabilities over the entire model pool for 4096 queries requires ≈20ms, which is orders of magnitude smaller than actual response generation time from language models (usually in seconds (Kwon et al., 2023)). This further strengthens the practicality of STARMAP for applications like routing, model ranking etc.

H IMPLEMENTATION DETAILS AND REPRODUCIBILITY CHECKLIST

We provide training code and data processing scripts for STARMAP in the supplementary submission. Across experiments in this work, STARMAP utilizes $r = 64$ learnable vectors for Latent bottleneck Attention Blocks (eq. (2)), and our default model embedding dimension is $d = 128$. We choose query, key and value dimensions same as d throughout all the attention layers. The feedforward network (FFN) is a two-layer MLP with hidden dimension $d_{\text{ff}} = 2d$. The tokenizer module h_ω is a single layer MLP with output dimension as d. Correctness predictor G_ψ is a 2-layer MLP with hidden dimension 64.

For training encoder-decoder pair (F_θ, G_ψ) in Section 2.3, we sample the query evaluations as input to the encoder $S_m^{\text{enc}} \subseteq S_m$ to generate model embeddings z_m, and independently sample a subset $S_m^{\text{dec}} \subseteq S_m$ for training the correctness predictor (S_m^{enc} and S_m^{dec} can have overlap, but only during training). The number of queries chosen in $(S_m^{\text{enc}}, S_m^{\text{dec}})$ can be varied to adjust to training hardware, our choices are detailed in the supplementary code submission. For all of the presented numerical experiments on correctness prediction and routing accuracy, test queries are only used for evaluating the decoder performance G_ψ, and never used for generating the embeddings $\{z_m\}_{m \in \mathcal{M}}$. This ensures that there is no leakage of test data into generating the model embeddings. Additionally, experiments examining correctness agreement rate utilize test set of common queries over which

model correctness agreement is evaluated (Figures 6 and 8), again ensuring no test queries used for generating embeddings and no training queries used for evaluation.

For the examined baselines, we utilize the same hyperparameters as provided in publicly available code implementation from respective authors.

I Additional Theoretical Analysis

In this section, we present a detailed theoretical analysis of STARMAP which supports and corroborates the empirical observations from Section 4, extending the theoretical examination presented in Section 3. Our theoretical contributions are directed along three aspects: (i) demonstrating the stability of model embeddings under partial evaluation sets, (ii) analyzing the robustness of query routing and model correctness prediction to varying query evaluation sets, and (iii) examining the generalization of model embedding generator to new models based on the number of models and query evaluation seen during training of the encoder-decoder architecture and number of test queries evaluated on a new model.

I.1 Assumptions and Notation

Below, we recall the notation used for our analysis as presented in Section 2. For a model $m \in \mathcal{M}$, let its query evaluation set be

$$S_m \;=\; \{(\phi(x_i),\, y^{(m)}(x_i))\}_{i=1}^{n_m}$$

where $\phi(x_i) \in \mathbb{R}^{d_\phi}$ is the query representation and $y_i^{(m)}(x_i) \in [0,1]$ is the observed evaluation score for model m on query x_i. Each query-evaluation pair is tokenized as

$$t_i^{(m)} = h_\omega\big(\phi(x_i), y_i^{(m)}\big) \in \mathbb{R}^d,$$

and encoder F_θ produces the model embedding

$$z_m = F_\theta(\mathcal{S}_m) \in \mathbb{R}^d.$$

The correctness predictor G_ψ then combines z_m with a query x to produce a score $G_\psi\big(z_m, \phi(x)\big)$ and the corresponding predicted correctness probability is

$$p_\psi(x, z_m) = \sigma\big(G_\psi\big(z_m, \phi(x)\big)\big),$$

where $\sigma(\cdot)$ denotes the sigmoid function.

When only a subset of k query evaluations is available for any model m, let

$$\widehat{\mathcal{S}}_{m,k} = \Big\{\big(\phi(x_i), y_i^{(m)}(x_i)\big)\Big\}_{i\in I_k}, \qquad |I_k| = k,$$

and define the partial-data embedding

$$\widehat{z}_{m,k} = F_\theta(\widehat{\mathcal{S}}_{m,k}).$$

The correctness predictor returns the logit $G_\psi(z, \phi(x))$ and the corresponding probability

$$p_\psi(x; z) = \sigma(G_\psi(z, \phi(x)))\,.$$

Assumption I.1 (Bounded evaluation tokens)**.** We assume that there exists a constant $B < \infty$ such that every query-model evaluation token satisfies

$$\|t_i^{(m)}\|_2 \le B.$$

We additionally express the stability of the encoder through the attention computation block. For a finite set of evaluation tokens $\mathcal{S}$, we denote

$$P_{\mathcal{S}} f = \frac{1}{|\mathcal{S}|} \sum_{t\in\mathcal{S}} f(t).$$

A latent (learned) query in the bottleneck attention block aggregates token values through an attention-weighted average of the form

$$\frac{P_{\mathcal{S}}\left[e^{s(q,t)}v(t)\right]}{P_{\mathcal{S}}\left[e^{s(q,t)}\right]},$$

where $s(q,t)$ is the attention logit and $v_j(t)$ is a coordinate of the value vector. The encoder combines these coordinate-wise outputs across its attention blocks to produce the model embedding. Let $\mathcal{F}_\theta$ encompass the scalar functions in the attention computation for each coordinate, over the latent (learned) queries and intermediate states under consideration.

Assumption I.2 (Stability of the encoder)**.** For the trained encoder F_θ, there is a finite constant C_θ, independent of the number of evaluations, such that

$$\|F_\theta(\mathcal{S}) - F_\theta(\mathcal{S}')\|_2 \le C_\theta \sup_{f\in\mathcal{F}_\theta} |P_{\mathcal{S}}f - P_{\mathcal{S}'}f|$$

for evaluation sets $\mathcal{S}$ and $\mathcal{S}'$ in the bounded domain under consideration. Every $f \in \mathcal{F}_\theta$ satisfies $|f(t)| \le U_\theta$. Moreover, $\mathcal{F}_\theta$ admits a finite η-cover in the uniform norm, of size $\mathcal{N}_\theta(\eta)$, for every $\eta > 0$.

For the fixed-depth latent-bottleneck attention based encoder, this bounds the effect of attention averages to the final generated model embedding. Along with bounded evaluation tokens (assumption I.1), assumption I.2 ensures that the generated model embeddings are finite and the effect of changing query evaluation set is bounded.

Assumption I.3 (Decoder stability)**.** For every query x, the correctness logit is L_G-Lipschitz in the model embedding:

$$|G_\psi(z,\phi(x)) - G_\psi(z',\phi(x))| \le L_G\|z - z'\|_2.$$

I.2 Stability of Embeddings to Evaluation Data Subsampling

We now quantify the convergence of a model embedding generated using k uniformly sampled (without replacement) query evaluations towards the full-data embedding generated using all n_m available evaluations for the model. The following theorem corroborates our empirical observations from Section 4.3 and Figure 5, indicating the sample efficiency and the robustness of model embeddings to the query evaluations used for generating them.

Theorem I.4 (Concentration of partial-data embeddings)**.** *Under assumptions I.1 and I.2, for* $1 \le k < n_m$*, any* $\eta > 0$*, and any* $\delta \in (0,1)$*, with probability at least* $1-\delta$*,*

$$\|\widehat{z}_{m,k} - z_m\|_2 \le \varepsilon_{m,k}(\delta;\eta),$$

where

$$\varepsilon_{m,k}(\delta;\eta) = C_\theta\left[2\eta + U_\theta\sqrt{\frac{2\rho_{n_m,k}}{k}\log\frac{2\mathcal{N}_\theta(\eta)}{\delta}}\right]$$

and

$$\rho_{N,k} = \begin{cases} 1 - \dfrac{k-1}{N}, & k \le N/2, \\ \left(1-\dfrac{k}{N}\right)\left(1+\dfrac{1}{k}\right), & k > N/2, \end{cases}$$

is the finite population correction term. For $k = n_m$*, the two embeddings are identical and we set* $\varepsilon_{m,n_m}(\delta) = 0$*. If* $\mathcal{N}_\theta(\eta)$ *grows polynomially in* $1/\eta$*, a suitable choice of* η *yields*

$$\|\widehat{z}_{m,k} - z_m\|_2 = \widetilde{\mathcal{O}}_\theta\left(\sqrt{\frac{\rho_{n_m,k}}{k}}\right).$$

Proof. Fix $f \in \mathcal{F}_\theta$. Since $|f(t)| \le U_\theta$, the values of f on $\mathcal{S}_m$ lie in an interval of length at most $2U_\theta$. The Hoeffding–Serfling inequality for sampling without replacement (Serfling, 1974; Bardenet & Maillard, 2015) therefore bounds $\left|P_{\widehat{\mathcal{S}}_{m,k}}f - P_{\mathcal{S}_m}f\right|$ with the finite-population correction $\rho_{n_m,k}$ as

$$\Pr\left(\left|P_{\widehat{\mathcal{S}}_{m,k}}f - P_{\mathcal{S}_m}f\right| > U_\theta\sqrt{\frac{2\rho_{n_m,k}}{k}\log\frac{2\mathcal{N}_\theta(\eta)}{\delta}}\right) \le \frac{\delta}{\mathcal{N}_\theta(\eta)}.$$

Applying this inequality to each function in an η-cover of $\mathcal{F}_\theta$, and further taking the union bound over $\mathcal{N}_\theta(\eta)$ results in every function satisfying the deviation bound with probability greater than $1-\delta$.

For any $f \in \mathcal{F}_\theta$, choose a cover element g with $\|f-g\|_\infty \le \eta$. Replacing f by g changes each of the two averages by at most η. Thus,

$$\sup_{f\in\mathcal{F}_\theta}\left|P_{\widehat{\mathcal{S}}_{m,k}}f - P_{\mathcal{S}_m}f\right| \le 2\eta + U_\theta\sqrt{\frac{2\rho_{n_m,k}}{k}\log\frac{2\mathcal{N}_\theta(\eta)}{\delta}}.$$

The embedding bound follows from assumption I.2. If $k = n_m$, the sampled set equals the full set, so their embeddings are identical. □

I.3 Robustness of Correctness Predictions and Routing Decisions

We next relate embedding stability to the correctness predictions used for routing. Because the predictor takes the model embedding as an input, assumption I.3 bounds the change in its logit, and the sigmoid gives a corresponding probability bound.

Proposition I.5 (Stability of correctness predictions)**.** *Under assumption I.3, for every query x and embeddings z, z',*

$$|G_\psi(z,\phi(x)) - G_\psi(z',\phi(x))| \le L_G\|z-z'\|_2,$$

and

$$|p_\psi(x;z) - p_\psi(x;z')| \le \frac{L_G}{4}\|z-z'\|_2.$$

Proof. The bound on output of the correctness predictor $(G_\psi(z,\phi(x))$ is from assumption I.3. For the probability bound, note that the sigmoid function is $1/4$-Lipschitz because

$$\sup_t \sigma'(t) = \sup_t \sigma(t)(1-\sigma(t)) = \frac{1}{4}.$$

Hence

$$|p_\psi(x;z) - p_\psi(x;z')| = |\sigma(G_\psi(z,\phi(x))) - \sigma(G_\psi(z',\phi(x)))| \le \frac{1}{4}|G_\psi(z,\phi(x)) - G_\psi(z',\phi(x))|,$$

and the result follows from the assumption I.3. □

Corollary I.6 (Correctness-prediction stability under subsampling)**.** *Under the conditions of theorem I.4 and proposition I.5, with probability at least $1-\delta$, simultaneously for every query x,*

$$|p_\psi(x;\widehat{z}_{m,k}) - p_\psi(x;z_m)| \le \frac{L_G}{4}\varepsilon_{m,k}(\delta;\eta).$$

For a query x, define the model routing choice (by using full-data embeddings) and its top-two score margin by

$$m^*(x) = \arg\max_{m\in\mathcal{M}} G_\psi(z_m,\phi(x))$$

and

$$\gamma(x) = G_\psi(z_{m^*(x)},\phi(x)) - \max_{j\neq m^*(x)} G_\psi(z_j,\phi(x)).$$

Theorem I.7 (Stability of routing decisions)**.** *Suppose that $\|\widehat{z}_m - z_m\|_2 \le \varepsilon_z$ for every $m \in \mathcal{M}$. Under assumption I.3, the routing decision on x is preserved whenever*

$$\gamma(x) > 2L_G\varepsilon_z.$$

Proof. Let $m^* = m^*(x)$. For any competing model $j \neq m^*$, the decoder stability condition (assumption I.3) gives

$$G_\psi(\widehat{z}_{m^*},\phi(x)) \ge G_\psi(z_{m^*},\phi(x)) - L_G\varepsilon_z$$

and

$$G_\psi(\widehat{z}_j,\phi(x)) \le G_\psi(z_j,\phi(x)) + L_G\varepsilon_z.$$

Therefore

$$\begin{aligned} G_\psi(\widehat{z}_{m^*},\phi(x)) - G_\psi(\widehat{z}_j,\phi(x)) &\ge G_\psi(z_{m^*},\phi(x)) - G_\psi(z_j,\phi(x)) - 2L_G\varepsilon_z \\ &\ge \gamma(x) - 2L_G\varepsilon_z. \end{aligned}$$

If $\gamma(x) > 2L_G\varepsilon_z$, the right-hand side is positive for every $j \neq m^*$, which implies that m^* remains the unique top-scoring (most suitable) model. □

Corollary I.8 (Sufficient evaluations for routing stability)**.** *Suppose each model's embedding is generated from a uniformly sampled subset of k evaluations. Set*

$$\varepsilon_k(\delta) = \max_{m \in \mathcal{M}} \varepsilon_{m,k}\left(\frac{\delta}{|\mathcal{M}|}; \eta\right).$$

With probability at least $1 - \delta$, every query satisfying

$$\gamma(x) > 2L_G \varepsilon_k(\delta)$$

has the same routing decision under the partial-data and full-data embeddings.

Proof. We invoke the theorem I.4 for each model, with the confidence level as $1 - \delta/|\mathcal{M}|$. The desired bound follows by applying the union bound and theorem I.7. □

I.4 Model Embedding Generation for New Models

In STARMAP, the encoder-decoder architecture is trained once on an initial pool of language models and their query evaluations, and generating embeddings for previously unseen models entails computing a forward pass over the encoder with the new model's query evaluation data. The suboptimality in estimating the model embedding for previously unseen models is dependent on the variety of the training data for the encoder and the number of query evaluations for the new model. The following theorem exemplifies the sample efficiency of STARMAP for generating embeddings of previously unseen models as described in Section 4.2 and Figure 3.

Let (x, y) be a test query and correctness label, not present in the evaluation set used for generating a new model's embedding. For the binary cross-entropy loss with logit g,

$$\ell(g, y) = \log(1 + e^g) - yg, \qquad y \in [0, 1].$$

Define

$$R_{\text{full}}(m) = \mathbb{E}_{(x,y)}\left[\ell(G_\psi(z_m, \phi(x)), y)\right],$$
$$R_k(m) = \mathbb{E}_{(x,y)}\left[\ell(G_\psi(\widehat{z}_{m,k}, \phi(x)), y)\right].$$

Let $R^*(m)$ be the optimal empirical risk under the test query distribution.

Theorem I.9 (New-model risk decomposition)**.** *Under the conditions of theorem I.4 and assumption I.3, with probability at least $1 - \delta$,*

$$R_k(m) - R^*(m) \le \underbrace{R_{\text{full}}(m) - R^*(m)}_{\textit{residual full-data error}} + L_G \varepsilon_{m,k}(\delta; \eta).$$

Proof. Add and subtract the full-data risk:

$$R_k(m) - R^*(m) = \big(R_k(m) - R_{\text{full}}(m)\big) + \big(R_{\text{full}}(m) - R^*(m)\big).$$

For $y \in [0, 1]$, binary cross-entropy is 1-Lipschitz with respect to its logit value, since

$$\left|\frac{\partial \ell(g, y)}{\partial g}\right| = |\sigma(g) - y| \le 1.$$

Consequently, assumption I.3 implies

$$|R_k(m) - R_{\text{full}}(m)| \le L_G \|\widehat{z}_{m,k} - z_m\|_2.$$

Applying theorem I.4 proves the desired bound. □

I.5 Online-Bandit Feedback Compatibility

Our experiments use an offline evaluation matrix, where each model is evaluated on a randomly sampled pool of queries. In a deployed routing system, however, feedback is typically online: for an incoming query x_t, the router selects one model m_t, observes only the reward or correctness r_t

of that selected model, and does not observe the counterfactual rewards of the models that were not selected. Thus, online logs have the form

$$\{(x_t, m_t, r_t, \mu_t(m_t \mid x_t))\}_{t=1}^{T},$$

where $\mu_t(m_t \mid x_t)$ is the probability with which the data-collection router selected m_t. This setting introduces selection bias: different models may be evaluated on different regions of the query distribution because of the previous routing policy. The role of propensity weighting is to correct this action-selection bias.

The most direct use of such query evaluation data is off-policy evaluation: given data collected by a deployed router, we may want to estimate how a different target routing policy π would have performed on the same query stream. This is possible when the logging policy has sufficient support.

Assumption I.10 (Support)**.** For a target routing policy $\pi(m \mid x)$, whenever $\pi(m \mid x) > 0$, one has $\mu_t(m \mid x) > 0$, where $\mu_t(\cdot|\cdot)$ denotes the data collection policy/router.

Proposition I.11 (Unbiased Inverse Propensity Scoring evaluation)**.** *For a deterministic target routing policy π, define*

$$\widehat{V}_{\mathrm{IPS}}(\pi) = \frac{1}{T}\sum_{t=1}^{T} \frac{\mathbf{1}\{m_t = \pi(x_t)\}}{\mu_t(m_t \mid x_t)}\, r_t.$$

Under assumption I.10,

$$\mathbb{E}\big[\widehat{V}_{\mathrm{IPS}}(\pi)\big] = V(\pi),$$

where $V(\pi)$ denotes the true expected value of policy π.

Proof. Condition on x_t. Then

$$\begin{aligned}\mathbb{E}\left[\frac{\mathbf{1}\{m_t = \pi(x_t)\}}{\mu_t(m_t \mid x_t)}\, r_t \;\middle|\; x_t\right] &= \sum_m \mu_t(m \mid x_t)\frac{\mathbf{1}\{m = \pi(x_t)\}}{\mu_t(m \mid x_t)}\mathbb{E}[r_t \mid x_t, m] \\ &= \mathbb{E}[r_t \mid x_t, \pi(x_t)].\end{aligned}$$

Taking expectation over x_t and averaging over t yields the claim. □

IPS (Inverse Propensity Scoring) keeps the model-query evaluations in which the data-collection router happened to choose the same model as the target router, and then upweights those rewards by the inverse probability of observing that action. Thus, an action selected with small probability represents a larger number of counterfactual cases in which that action could have been selected but was not. This gives an unbiased estimate under correct propensities and support, although very small propensities can increase variance.

A lower-variance alternative is the doubly robust evaluation, which combines propensity weighting with a learned reward model. In our setting, the correctness predictor naturally provides such a reward model.

Proposition I.12 (Doubly robust evaluation)**.** *Let $\widehat{r}(x, m)$ be a reward model, which in our setting can be instantiated by the* STARMAP *correctness predictor. Define*

$$\widehat{V}_{\mathrm{DR}}(\pi) = \frac{1}{T}\sum_{t=1}^{T}\left[\widehat{r}(x_t, \pi(x_t)) + \frac{\mathbf{1}\{m_t = \pi(x_t)\}}{\mu_t(m_t \mid x_t)}\big(r_t - \widehat{r}(x_t, m_t)\big)\right].$$

If the logging propensities are correct, then $\widehat{V}_{\mathrm{DR}}(\pi)$ is unbiased. Moreover, if the reward model $\widehat{r}(x, a)$ is correct, then $\widehat{V}_{\mathrm{DR}}(\pi)$ is unbiased even if the propensities are misspecified.

Proof. Condition on x_t. If the propensities are correct,

$$\begin{aligned}&\mathbb{E}\left[\widehat{r}(x_t, \pi(x_t)) + \frac{\mathbf{1}\{m_t = \pi(x_t)\}}{\mu_t(m_t \mid x_t)}\big(r_t - \widehat{r}(x_t, m_t)\big) \;\middle|\; x_t\right] \\ &= \widehat{r}(x_t, \pi(x_t)) + \mathbb{E}[r_t - \widehat{r}(x_t, \pi(x_t)) \mid x_t, \pi(x_t)] \\ &= \mathbb{E}[r_t \mid x_t, \pi(x_t)].\end{aligned}$$

This proves unbiasedness under correct propensities.

If instead the reward model is correct, then

$$\mathbb{E}[r_t - \widehat{r}(x_t, m_t) \mid x_t, m_t] = 0,$$

so the correction term has zero conditional expectation regardless of the denominator, and

$$\mathbb{E}\big[\widehat{V}_{\mathrm{DR}}(\pi) \mid x_t\big] = \widehat{r}(x_t, \pi(x_t)) = \mathbb{E}[r_t \mid x_t, \pi(x_t)].$$

Taking expectation over x_t and averaging over t completes the proof. □

The same evaluation data-collection propensities can also be used for learning from bandit feedback. In the full-information setting, STARMAP can be trained on losses for many model-query pairs. In the online bandit setting, only the selected action (model choice) is labeled, so a naive loss over observed rewards would overrepresent the models and query regions favored by the evaluation data collection router. This selection bias can be removed by replacing the full-information loss with an inverse-propensity-weighted observed-action loss: each observed reward contributes to training in proportion to how unlikely it was to be observed under the logging policy. This gives an unbiased surrogate for the corresponding full-information objective over actions, under the same support and correct-propensity assumptions. In practice, clipped or self-normalized weights can be used to control variance when propensities are small.

The same issue arises when constructing embeddings from query-model evaluation data collected in an online manner. If the embedding for model m is built only from the queries that the deployed router happened to send to m, then the resulting embedding may represent model m's behavior on a biased slice of the query distribution. To recover an embedding corresponding to a desired target query distribution, the query-evaluation tokens for each model can be reweighted by the inverse probability that the model was selected for generating responses to those queries, optionally combined with query-distribution weights if the deployment query stream differs from the target evaluation distribution.

Thus, bandit feedback and online evaluation data collection can be accommodated with STARMAP. Inverse Propensity Scoring and doubly robust estimators provide standard tools for off-policy evaluation of routing policies, propensity-weighted losses provide a corresponding route for training under selected-action feedback, and weighted token aggregation can debias model embeddings constructed from non-uniform online evaluations.