Low Dimensional Embeddings for Model Capability Understanding

Shivam Patel [1] **William Cocke** [1] **Gauri Joshi** [1]

Abstract

The rapidly growing ecosystem of Large Language Models (LLMs) makes it increasingly difficult to manage and utilize the expanding model pool. We propose LOCUS, an attention-based method that produces low-dimensional embeddings capturing a model's capabilities across queries. LOCUS deterministically generates embeddings from query encodings and evaluation scores via a forward pass, enabling new models to be added and existing embeddings to be refined without retraining. A correctness predictor built on these embeddings achieves state-of-the-art routing accuracy on unseen queries. Experiments show that LOCUS requires up to $4.8\times$ fewer query evaluations than baselines while producing robust, geometrically meaningful embeddings whose proximity reflects model similarity, supporting model comparison, clustering, portfolio selection, and proxying unavailable models.

1. Introduction

Large Language Models (LLMs) have been utilized for a wide range of tasks (OpenAI, 2024; Brown et al., 2020), 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 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 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

[1]Carnegie Mellon University. Correspondence to: Shivam Patel <shivamap@andrew.cmu.edu>.

ICML 2026 *AdaptFM* Workshop on Resource-Adaptive Foundation Model Inference.

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

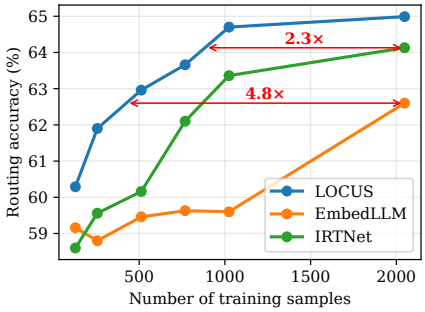


Figure 1. **Routing accuracy vs number of training samples.** LOCUS uses $\mathbf{2.3-4.8}\times$ fewer training samples (number of queries on which each model is evaluated) as compared to baselines, while maintaining high routing accuracy performance.

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.

Related works. Prior work on model embeddings can be broadly divided into parametric and nonparametric approaches. Parametric methods learn *per-model embeddings as trainable parameters* together with a correctness predictor trained on model–query evaluations. EmbedLLM (Zhuang et al., 2025) follows this design, while IRT-Net (Chen et al., 2025) and JE-IRT (Yao et al., 2025) adopt item response theory for interpretable prediction (Hambleton et al., 1991); a related matrix-factorization approach represents models in query-encoding space (Kashani et al., 2025). Although effective for correctness prediction and routing, these embeddings are optimized by SGD and are not uniquely determined by the evaluation data, making geometry-based analyses such as nearest neighbors, clustering, and similarity search sensitive to training randomness and overparameterization; see Appendix F. In contrast, nonparametric methods compute embeddings directly from shared evaluations using geometric operations. For example, LLM-DNA (Wu et al., 2025) concatenates response encodings and applies a Gaussian random projection, but requires all models to be evaluated on the same queries and cannot naturally exploit partial evaluations or additional evaluations to refine embeddings.

The remainder of the paper is organized as follows: Section 2 introduces LOCUS (**LO**w-dimensional embeddings for Model **C**apability **U**nder**S**tanding); Section 3 reports results and analyses; and Section 4 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).

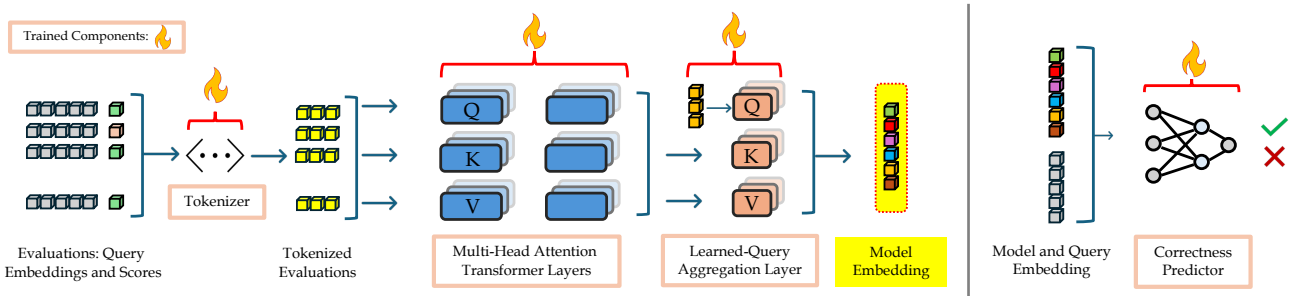
2.1. Attention-based model capability encoder F_θ

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 to form

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

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 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



(a) Model Embedding Generator F_θ *(b)* Correctness Predictor G_ψ

Figure 2. **Overview of LOCUS.** 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 (i.e., score) of chosen model on the query.

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 $d \to d_{\text{ff}} \to d$) 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 is:

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

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 $H_m^{(\ell)}$:

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

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:

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

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

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

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 $\widehat{p}^{(m)}(x) \triangleq \widehat{p}_\psi\left(y^{(m)}(x)=1\middle|z_m,\phi(x)\right)$. 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. Experiments

Models, tasks, and evaluation sets. We evaluate LOCUS on 112 language models (Zhuang et al., 2025) spanning diverse sizes and capabilities, including general and finetuned models. Queries are sampled from 10 public binary-scored benchmarks covering tasks such as math, general knowledge, and reasoning; continuous-score evaluations are examined in Appendix B.1. We use `all-mpnet-base-v2` (Song et al., 2020) as the sentence encoder $\phi(x)$ for query representations, with details on models and datasets in Appendix A and encoder ablations in Appendix C.

Table 1. Overall routing and correctness prediction accuracy (%) across approaches and training-set sizes (number of evaluation queries). LOCUS outperforms examined baselines on routing accuracy, and all approaches perform comparatively on correctness prediction accuracy. Extended results presented in Appendix B. **Best** and second best emphasized in each case.

Approach (# Query Evaluations)	Routing Accuracy (%)			Corr. Pred Acc (%)		
	256	512	1024	256	512	1024
LOCUS (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**

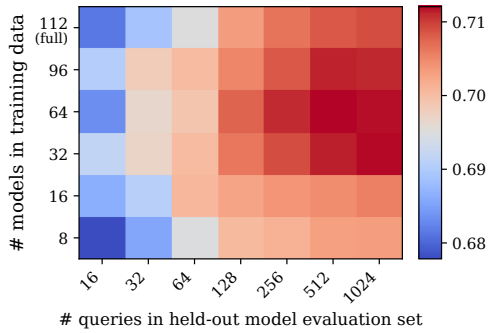


Figure 3. **New LLM onboarding.** Correctness prediction accuracy on 16 held-out models: with varying number of query evaluations for embedding held-out models and varying number of models present in training data (top row indicates (F_θ, G_ψ) trained on the full set of all language models). Only a few queries (≈128) are required to generate informative embeddings for unseen models, and there is a negligible ($< 1\%$) accuracy gap between using an encoder trained on the full (top row) versus a partial set of models.

Unless stated otherwise, our attention-based encoder is trained on 4096 query evaluations per language model and uses two latent bottleneck attention blocks (eq. (2)) with $L = 2$, 4-headed MHA sublayers, $r = 64$ learned latent vectors per block, and an attention-based aggregation layer (eq. (3)). The correctness predictor (eq. (4)) is a single-hidden-layer MLP with $d_{\text{hidden}} = 64$, and the model embedding dimension is $d = 128$; hyperparameter ablations are in Appendix D. Test queries are used only for evaluating correctness prediction and model performance agreement, never for generating model embeddings; see Appendix H.

Baselines. We compare against gradient-trained embedding baselines from query evaluation data: **EmbedLLM** (Zhuang et al., 2025) and **IRT-Net** (Chen et al., 2025). Comparisons with nonparametric methods, including LLM-DNA (Wu et al., 2025) and kNN router (Jitkrittum et al., 2025), are deferred to Appendix B.2.

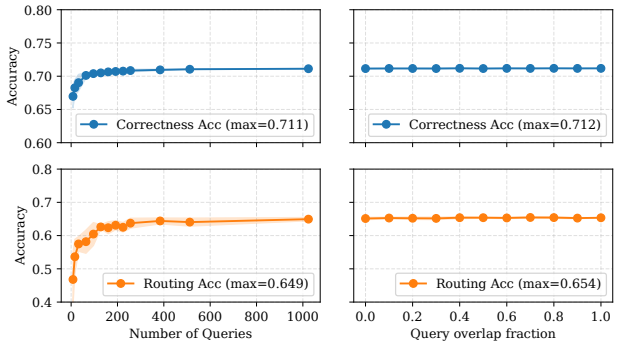


Figure 4. **Robustness to Varying Query Evaluations.** Correctness prediction accuracy (top) and routing accuracy (bottom) versus (left) the number of evaluations used to construct embeddings and (right) overlap fraction with a reference evaluation set. Performance saturates with 128-256 evaluations and remains stable across overlap fractions and query subsampling, showing robustness to evaluation queries.

3.1. 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 obtain binary predictions by thresholding, $\widehat{y}^{(m)}(x) = \mathbb{I}\{\widehat{p}^{(m)}(x) \geq 0.5\}$, and report correctness prediction accuracy as the average agreement $\mathbb{I}\{\widehat{y}^{(m)}(x) = y^{(m)}(x)\}$ over all model–query pairs. Routing accuracy is computed by selecting, for each test query, the model with the highest predicted correctness probability and checking whether its response is correct. Across training set sizes, LOCUS outperforms the examined baselines in both routing and correctness prediction accuracy, as shown in Figure 1 and Table 1. We attribute this to attention blocks extracting model capability patterns directly from query evaluations, rather than relying on backpropagation-learned embeddings to describe observed correctness patterns. Additional numerical results and ablations are provided in Appendices B to D, with wall-clock time complexity in Appendix G.

3.2. 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.

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 $\in \{8, 16, \ldots, 92\}$ (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.

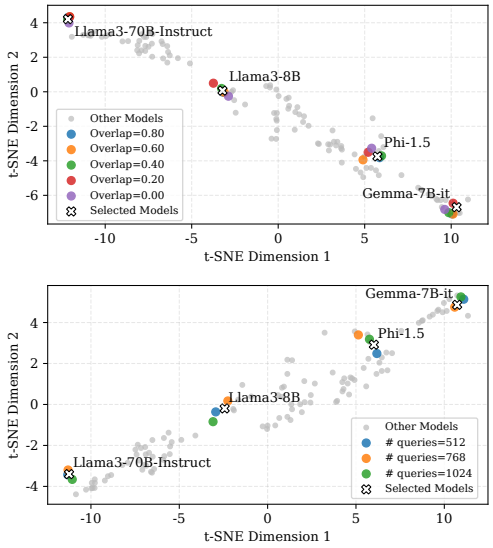


Figure 5. **Robustness to varying query evaluations.** Embeddings for selected models recomputed from evaluation sets with (top) varying overlap with the reference set and (bottom) subsampled queries from the reference set, visualized with t-SNE. Gray points denote other models in the pool.

3.3. Robustness to Varying Query Evaluations

We test sensitivity to the evaluation set given to a pretrained encoder F_θ trained on the full model pool by first computing reference embeddings z_m^{ref} from a fixed set of size n_{ref} (e.g., 4096), then recomputing embeddings after changing **which** queries are used at fixed size or **how many** are used via subsampling $n \in \{64, 128, 256, \ldots, n_{\text{ref}}\}$. Correctness prediction and routing accuracy remain stable across overlap fractions $\alpha \in [0, 1]$ with the reference set, and performance saturates with only $\approx$ 128–256 evaluations (Figure 4); t-SNE projections further show that regenerated embeddings stay close to z_m^{ref} and remain geometrically stable under both perturbations (Figure 5) (van der Maaten & Hinton, 2008). This robustness also provides a preliminary fingerprinting signal: if a new model consistently embeds near the same reference model across independently sampled evaluation sets, it may indicate model duplication (Appendix E.3 for additional experiments) (Zeng et al., 2024; Tsai et al., 2025).

3.4. Embedding Geometry Reflects Similarity

For each model pair $(m_1, m_2) \in \mathcal{M} \times \mathcal{M}$, we compare embedding-space distance $d_z(z_{m_1}, z_{m_2})$ using cosine and Euclidean distances with correctness *disagreement rate* on a common test set, defined as the fraction of queries where binary correctness labels differ. Figure 6 shows strong posi-

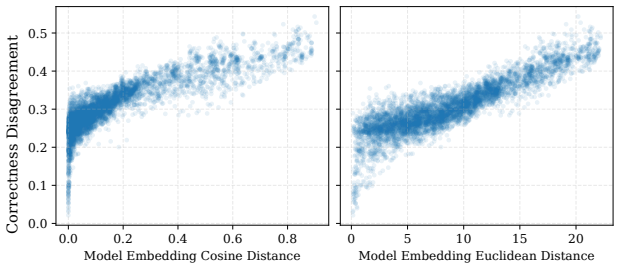


Figure 6. **Embedding distance versus correctness disagreement for model pairs.** Embedding distances strongly correlate with disagreement in observed correctness outcomes (Pearson 0.85 for cosine distance and 0.89 for Euclidean distance), indicating that embedding geometry reflects model similarity.

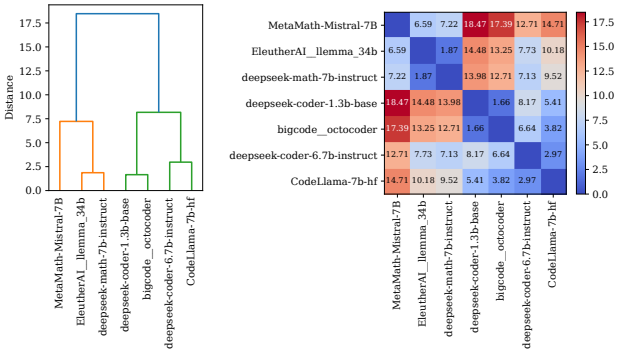


Figure 7. **Identifying model families.** Hierarchical clustering on model embeddings separates model families, shown here for math (orange) and code (green) families.

tive correlation between embedding distance and disagreement, with Pearson correlations of 0.845 (cosine) and 0.887 (Euclidean), and high rank correlations (Spearman ≈ 0.88, Kendall ≈ 0.71), indicating that embedding distances capture meaningful model similarity. Hierarchical clustering on Euclidean embedding distances (Cohen-addad et al., 2019) further recovers coherent groups aligned with known model specializations, such as math- and code-finetuned models (Figure 7), showing that the geometry captures shared capability structure beyond individual models.

3.5. Practical Utilities Enabled by Model Embeddings

Nearest neighbors as proxies under model unavailability. When a preferred model is unavailable, nearby models in embedding space can serve as capability-preserving *proxies*. Ranking neighbors by embedding distance, Figure 8 shows that the closest neighbor achieves **79%** average correctness agreement, with agreement decreasing by larger k.

Model portfolio selection from embeddings. Embeddings also enable compact portfolios that reduce redundancy while preserving routing accuracy. For count-constrained portfolios, `k-center` minimizes the maximum distance to the portfolio (Gonzalez, 1985), while `k-medoids` minimizes average distance (Kaufman & Rousseeuw, 1990); a **15-model portfolio matches full-pool routing accuracy** over all 112 models (Appendix E.4.2). Under parameter budgets, `coverage-greedy` covers the embedding space while satisfying the budget and achieves strong routing performance with far fewer than the full pool's 1930B parameters (Appendix E.4.3). Additional utilities in Appendix E.

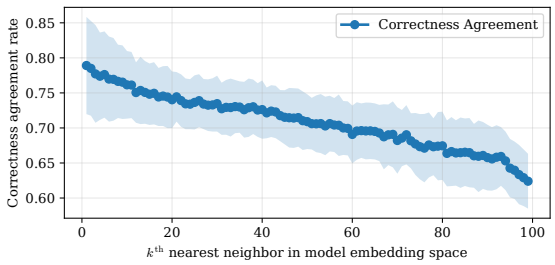


Figure 8. **Nearest-neighbor model proxies.** Average correctness agreement between each model and its k-th nearest neighbor in embedding space. Closest neighbors agree most, with gradual decay for farther neighbors.

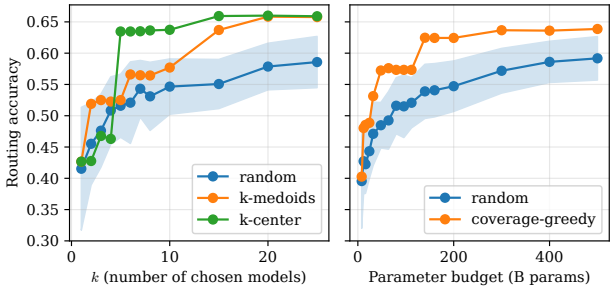


Figure 9. **Model Portfolio Selection.** Embeddings guide model subset selection for routing. *Left:* `k-center` and `k-medoids` maximize coverage under a model-count budget. *Right:* `coverage-greedy` maximizes coverage while prioritizing smaller models under a parameter budget.

4. Concluding Remarks

In this paper, we introduced LOCUS, a novel attention-based framework that embeds Large Language Models into a low-dimensional space based on inference capabilities. Unlike prior approaches, LOCUS enables training-free onboarding of new models without altering existing embeddings, while requiring up to $4.8\times$ fewer query evaluations to produce robust representations. Beyond performance prediction, the learned geometry captures behavioral similarity, supporting downstream utilities such as model portfolio selection and resilient fallback routing under model unavailability. As proprietary and open-source model ecosystems grow, capability-oriented embeddings provide a scalable, geometrically meaningful foundation for efficient model management and discovery. We leave extensions to multimodal models and adaptive query evaluation for future work.

Acknowledgments

This work was partially supported by NSF grants CCF 2045694, CNS-2112471, CPS-2111751, ONR grant N00014-23-1-2149, and an AI2C Seed grant. This work used Bridges-2 GPU at the Pittsburgh Supercomputing Center through allocation CIS250429 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296 (Boerner et al., 2023).

References

Amini, A., Gabriel, S., Lin, S., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. 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.

Bisk, Y., Zellers, R., Le bras, R., Gao, J., and Choi, Y. 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.

Boerner, T. J., Deems, S., Furlani, T. R., Knuth, S. L., and Towns, J. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, PEARC '23, pp. 173–176, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399852. doi: 10.1145/3569951.3597559. URL https://doi.org/10.1145/3569951.3597559.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (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.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., and Xie, X. 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.

Chen, J., Wang, C., Zhang, G., Ye, P., Bai, L., Hu, W., Qu, Y., and Hu, S. Learning compact representations of llm abilities via item response theory, 2025. URL https://arxiv.org/abs/2510.00844.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M. I., Gonzalez, J. E., and Stoica, I. 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.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Cohen-addad, V., Kanade, V., Mallmann-trenn, F., and Mathieu, C. 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.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., and Awadallah, A. H. 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.

Fix, E. and Hodges, J. L. 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.

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

Gonzalez, T. F. 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.

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

Hambleton, R. K., Swaminathan, H., and Rogers, H. J. *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.

Hartigan, J. A. and Wong, M. A. 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.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Horwitz, E., Kurer, N., Kahana, J., Amar, L., and Hoshen, Y. 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.

Hu, E. J., yelong shen, Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*, 2024. URL https://openreview.net/forum?id=IVXmV8Uxwh.

Ilse, M., Tomczak, J., and Welling, M. Attention-based deep multiple instance learning. In Dy, J. and Krause, A. (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.

Jaegle, A., Gimeno, F., Brock, A., Zisserman, A., Vinyals, O., and Carreira, J. Perceiver: General perception with iterative attention, 2021. URL https://arxiv.org/abs/2103.03206.

Jitkrittum, W., Narasimhan, H., Rawat, A. S., Juneja, J., Wang, C., Wang, Z., Go, A., Lee, C.-Y., Shenoy, P., Panigrahy, R., Menon, A. K., and Kumar, S. Universal model routing for efficient llm inference, 2025. URL https://arxiv.org/abs/2502.08773.

Kashani, I., Mendelson, A., and Nemcovsky, Y. Representing LLMs in prompt semantic task space. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (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/.

Kaufman, L. and Rousseeuw, P. J. 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.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In Chaudhuri, K. and Salakhutdinov, R. (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.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. 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.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. 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.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Li, D., Jiang, B., Huang, L., Beigi, A., Zhao, C., Tan, Z., Bhattacharjee, A., Jiang, Y., Chen, C., Wu, T., Shu, K., Cheng, L., and Liu, H. From generation to judgment: Opportunities and challenges of LLM-as-a-judge. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (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/.

Lin, S., Hilton, J., and Evans, O. TruthfulQA: Measuring how models mimic human falsehoods. In Muresan, S., Nakov, P., and Villavicencio, A. (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/.

Lin, T., Wang, Y., Liu, X., and Qiu, X. A survey of transformers, 2021. URL https://arxiv.org/abs/2106.04554.

Lin, Z., Feng, M., dos Santos, C. N., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. A structured self-attentive sentence embedding, 2017. URL https://arxiv.org/abs/1703.03130.

Liu, J., Cui, L., Liu, H., Huang, D., Wang, Y., and Zhang, Y. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In Bessiere, C. (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.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. Object-centric learning with slot attention, 2020. URL https://arxiv.org/abs/2006.15055.

Miao, S.-y., Liang, C.-C., and Su, K.-Y. A diverse corpus for evaluating and developing English math word problem solvers. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (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.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=8sSqNntaMr.

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

Pal, A., Umapathi, L. K., and Sankarasubbu, M. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Flores, G., Chen, G. H., Pollard, T., Ho, J. C., and Naumann, T. (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.

Patel, S., Jali, N., Mallick, A., and Joshi, G. Proxrouter: Proximity-weighted llm query routing for improved robustness to outliers, 2025. URL https://arxiv.org/abs/2510.09852.

Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J., and Bowman, S. R. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=Ti67584b98.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Sap, M., Rashkin, H., Chen, D., Le Bras, R., and Choi, Y. Social IQa: Commonsense reasoning about social interactions. In Inui, K., Jiang, J., Ng, V., and Wan, X. (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/.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. Mpnet: Masked and permuted pre-training for language understanding. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (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.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. 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.

Tsai, Y.-Y., Guo, C., Yang, J., and van der Maaten, L. Rofl: Robust fingerprinting of language models, 2025. URL https://arxiv.org/abs/2505.12682.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (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.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (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.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Wu, Z., Zhao, H., Wang, Z., Guo, J., Wang, Q., and He, B. Llm dna: Tracing model evolution via functional representations, 2025. URL https://arxiv.org/abs/2509.24496.

Xu, X., Li, M., Tao, C., Shen, T., Cheng, R., Li, J., Xu, C., Tao, D., and Zhou, T. A survey on knowledge distillation of large language models. *CoRR*, abs/2402.13116, 2024. URL https://doi.org/10.48550/arXiv.2402.13116.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In Knight, K., Nenkova, A., and Rambow, O. (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/.

Yao, L. H., Jarvis, N., Zhan, T., Ghosh, S., Liu, L., and Jiang, T. Je-irt: A geometric lens on llm abilities through joint embedding item response theory, 2025. URL https://arxiv.org/abs/2509.22888.

Zeng, B., Wang, L., Hu, Y., Xu, Y., Zhou, C., Wang, X., Yu, Y., and Lin, Z. 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.

Zhang, Y., Li, H., Wang, C., Chen, L., Zhang, Q., Ye, P., Feng, S., Wang, D., Wang, Z., Wang, X., Xu, J., Bai, L., Ouyang, W., and Hu, S. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants, 2025. URL https://arxiv.org/abs/2505.19797.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. A survey of large language models, 2025. URL https://arxiv.org/abs/2303.18223.

Zhuang, R., Wu, T., Wen, Z., Li, A., Jiao, J., and Ramchandran, K. Embedllm: Learning compact representations of large language models. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R. (eds.), *International Conference on Representation Learning*, volume 2025, pp. 76913–76926, 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 3.2 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`.

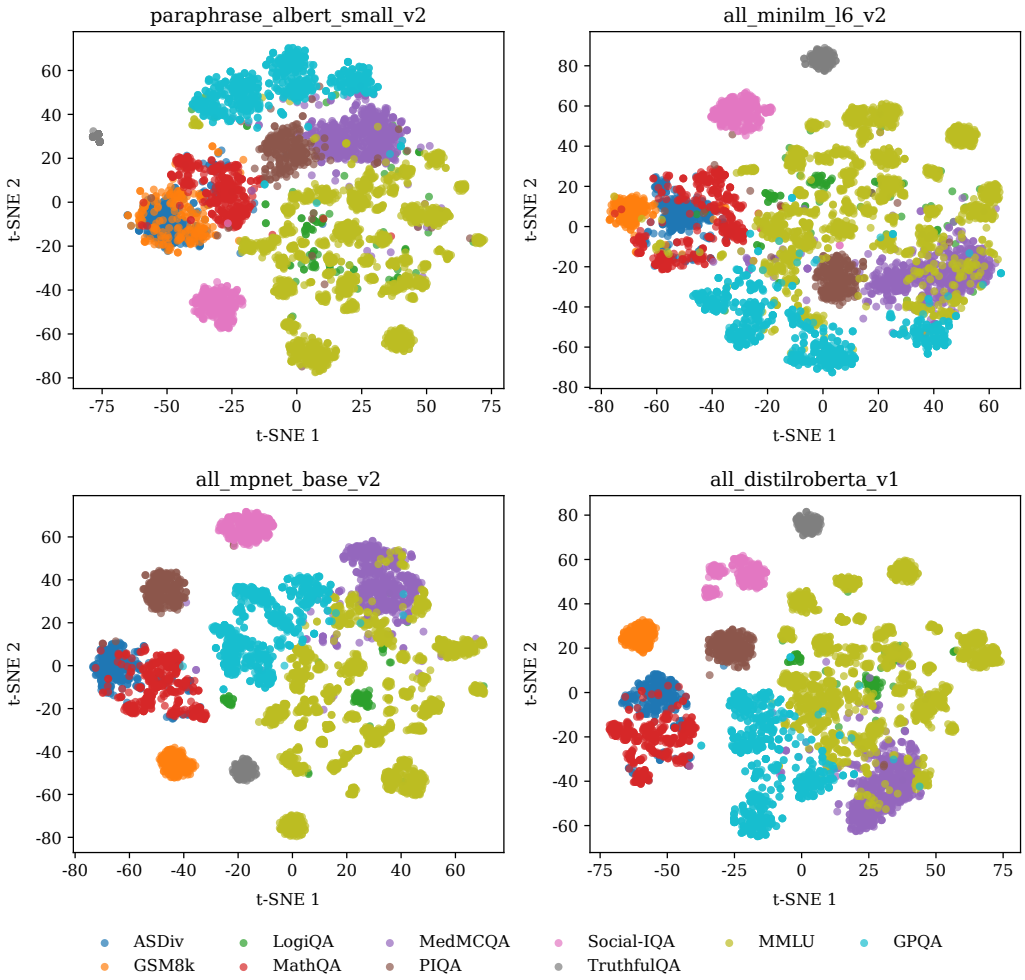


Figure 10. Query encodings visualized using t-SNE (van der Maaten & Hinton, 2008), colored by dataset. Queries from the same dataset represent similar tasks, and their sentence encodings are placed close by in the encoding space. Plots presented for four different sentence encoders `all_mpnet_base_v2` (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 LOCUS on Continuous Scoring Tasks

In the main empirical analysis presented in Section 3, 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 LOCUS beyond binary scoring approaches.

Table 3. Overall routing and correctness prediction accuracy (%) 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
LOCUS (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 LOCUS, 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 nonparameteric 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, LOCUS 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
LOCUS (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 LOCUS 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 LOCUS with LLM-DNA on ProxRouter (Patel et al., 2025) data. We observe that LOCUS outperforms LLM-DNA on routing accuracy and correctness prediction over all number of queries used for generating the model embeddings.

C. Ablation on choice of Sentence Encoder for Query Representation

In this section, we present correctness prediction accuracy and routing accuracy of LOCUS 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 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
LOCUS (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

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 LOCUS (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 LOCUS 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 LOCUS 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 LOCUS 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.

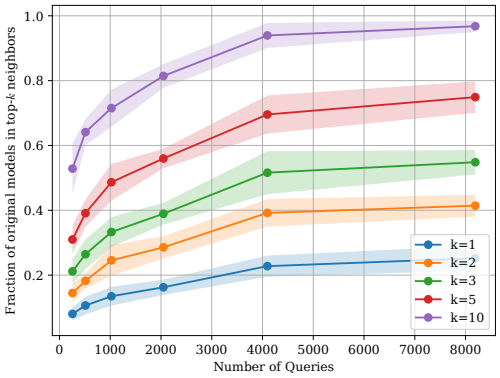


Figure 11. We obtain model embeddings based on random assignment of correctness labels to queries to match dataset-wise accuracy scores of actual LLMs in the pool. Then we find the proportion of original model embeddings that are among the $(1, 2, 3, 5, 10)$ nearest neighbors of the new generated embedding.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

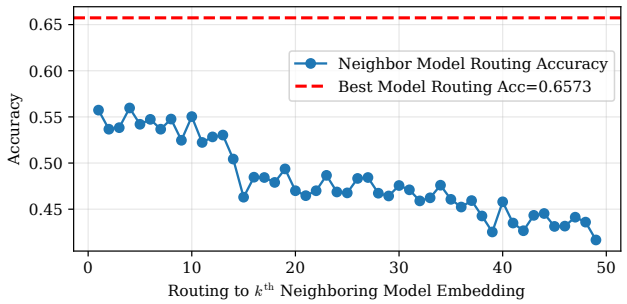


Figure 12. **Resilient Routing to Fallback Models.** Selected models for query routing might not always be available due to congestion, node failure etc. Routing to next closest model embedding provides a fallback mechanism, while maintaining up to 85% of overall routing performance. Performance gracefully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

Table 10. Overall routing and correctness prediction accuracy (%) for LOCUS with varying number of Attention blocks 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

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 k^{th} 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.

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.

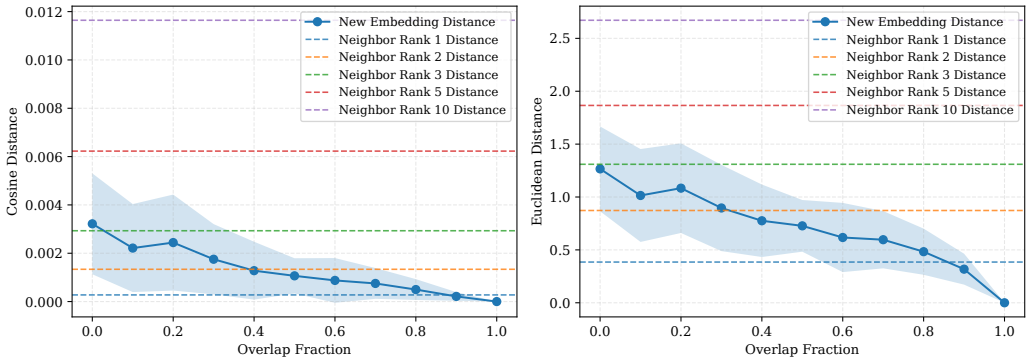


Figure 13. Distance between new generated model embedding to reference embedding (for CodeLlama-13b-hf model) w.r.t. changing the overlap fraction of queries in evaluation set used for generating model embeddings. Horizontal dashed lines denote distance of chosen model's reference embedding to other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

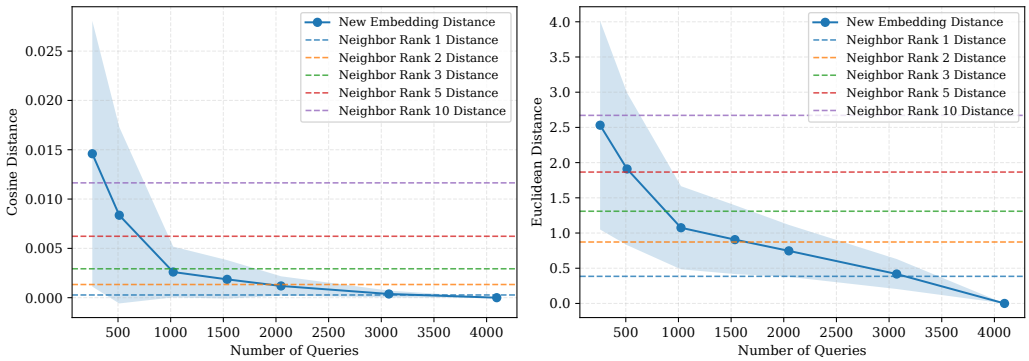


Figure 14. Distance between new generated model embedding to reference embedding (for CodeLlama-13b-hf model) w.r.t. changing the number of queries in evaluation set used for generating model embeddings. Horizontal dashed lines denote distance of chosen model's reference embedding to other models' embeddings.

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 $\text{Acc}(\mathcal{M}_{\text{sel}})$ 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

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

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:

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

and stop when no feasible model remains.

Evaluation. For each parameter count constraint $\mathcal{C}$, we select $\mathcal{M}^{\mathcal{C}}_{\text{sel}}$ using eq. (16) and evaluate $\text{Acc}(\mathcal{M}^{\mathcal{C}}_{\text{sel}})$ by restricting evaluation to models present in $\mathcal{M}^{\mathcal{C}}_{\text{sel}}$. 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 ≈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 $G_\psi(e_\theta^{(m)}, x) = \mathrm{MLP}_\psi(e_\theta^{(m)} \odot W_\psi x)$, 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:

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

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 $\{e_{\theta_{\mathrm{re}}}^{(m)}\}_{m\in\mathcal{M}}$ with the same model query evaluation data. Mathematically, we perform:

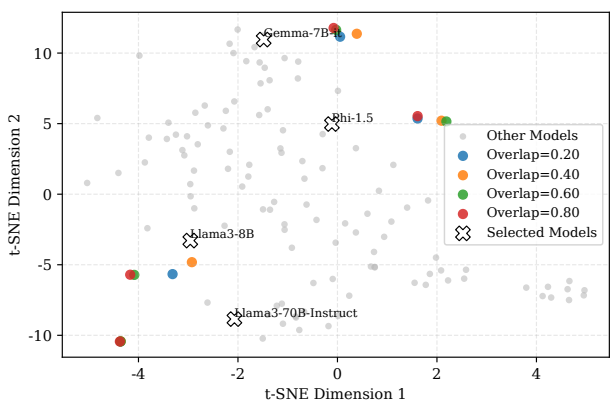
$$\min_{\theta_{\mathrm{re}}} \quad \mathbb{E}_m\, \mathbb{E}_x \quad \Big[\mathrm{BCE}\big(G_\psi(e_{\theta_{\mathrm{re}}}^{(m)}, x),\ y^{(m)}(x)\big)\Big] \tag{18}$$

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 $\{e_{\theta_{\mathrm{re}}}^{(m)}\}_{m\in\mathcal{M}}$ with originally trained embeddings $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ for similarity.

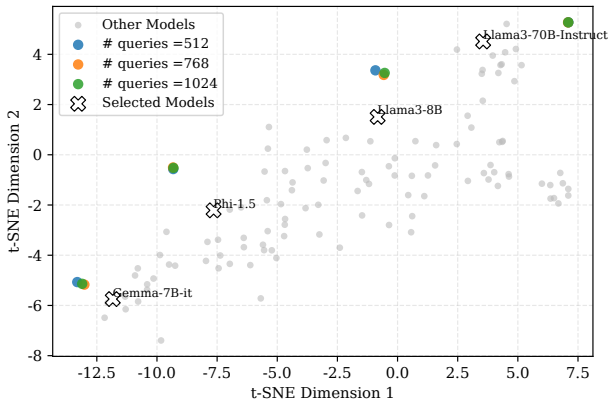
Because the training objective in eq. (18) differs from that in eq. (17), we anticipate small changes in the regenerated model embeddings, even though the embeddings are learned on the same query evaluation data. However, they should remain close to the original embeddings to preserve the practically desirable properties described in Section 1. If the regenerated embeddings $\{e_{\theta_{\mathrm{re}}}^{(m)}\}_{m\in\mathcal{M}}$ 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 3 infeasible.

We examine EmbedLLM (Zhuang et al., 2025) through a similar approach as described in Section 3.3, 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 3.3), 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.

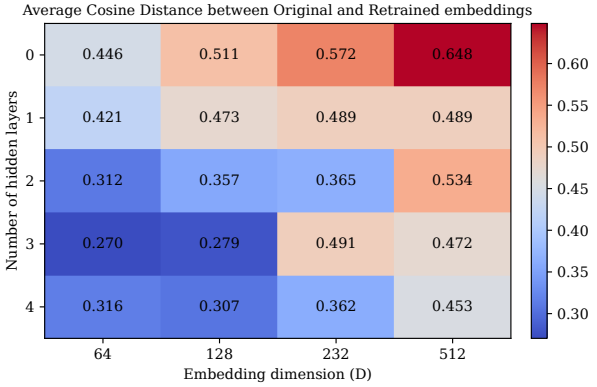


Figure 16. Average cosine distance between initially trained model embeddings $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ and regenerated model embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m\in\mathcal{M}}$. We present results across number of hidden layers in MLP_ψ (with $d_{\text{hidden}} = 64$), along 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.

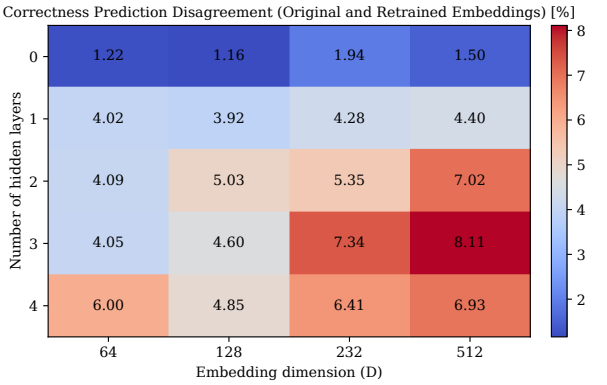
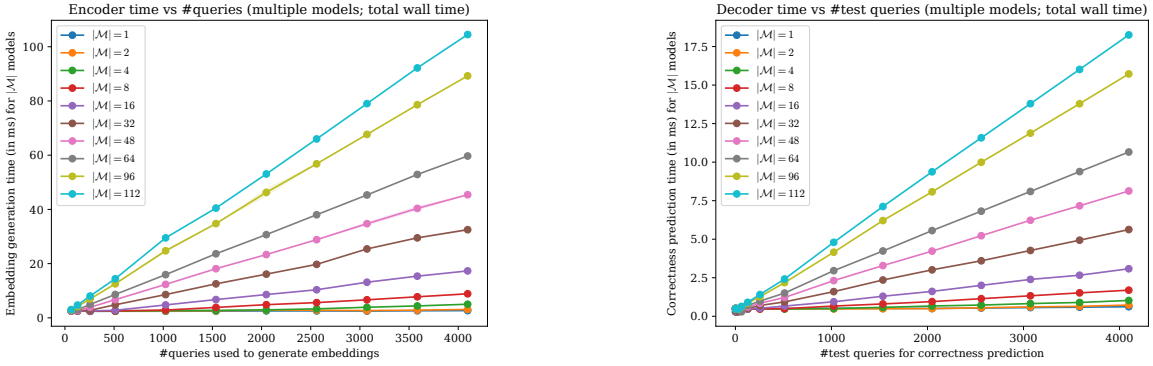


Figure 17. Query Correctness Disagreement between initially trained model embeddings $\{e_\theta^{(m)}\}_{m\in\mathcal{M}}$ and regenerated model embeddings $\{e_{\theta_{\text{re}}}^{(m)}\}_{m\in\mathcal{M}}$. We present the absolute percentage disagreement on correctness prediction averaged over all 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 nonidentical capability representation, even if trained on the same query evaluation data.

G. Time Complexity of LOCUS for Generating Embeddings and Correctness Prediction

In this section, we examine the time complexity of LOCUS 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 LOCUS (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 LOCUS 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 LOCUS for applications like routing, model ranking etc.

H. Implementation Details and Reproducibility Checklist

We provide training code and data processing scripts for LOCUS in the supplementary submission. Across experiments in this work, LOCUS 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.