UNIVERSAL LLM ROUTING WITH CORRECTNESS-BASED REPRESENTATION

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, Sanjiv Kumar

{wittawat, hnarasimhan, ankitsrawat, jeeveshjuneja, zifengw, chenyulee, shenoypradeep, rinap, adityakmenon, sanjivk}@google.com

Google

ABSTRACT

Large language models' significant advances in capabilities are accompanied by significant increases in inference costs. *Model routing* is a simple technique for reducing inference cost, wherein one maintains a pool of candidate LLMs, and learns to route each prompt to the smallest feasible LLM. Existing works focus on learning a router for a *fixed* pool of LLMs. In this paper, we consider the problem of *dynamic* routing, where *new, previously unobserved* LLMs are available at test time. We propose a new approach to this problem that relies on representing each LLM as a *feature vector*, derived based on predictions on a set of representative prompts. Based on this, we detail an effective strategy relying on cluster-based routing. We prove that the strategy is an estimate of a theoretically optimal routing rule. Experiments on a range of public benchmarks show the effectiveness of the proposal in routing amongst more than 30 unseen LLMs.

1 INTRODUCTION

Advances in capabilities of large language models (LLMs) come with an increasing inference cost. Our interest is in *model routing* for efficient inference. Here, one maintains a pool of candidate LLMs of various sizes and capabilities. Given a query, one learns to predict the lowest-cost LLM which can reasonably address the query. In doing so, one can learn to use high-cost LLMs sparingly, only on the (relatively) few "hard" inputs. This is a conceptually simple, but highly effective technique which has seen a surge of recent interest (Hendy et al., 2023; Hari & Thomson, 2023; Ding et al., 2024; Šakota et al., 2024; Chen et al., 2024b; Hu et al., 2024; Shnitzer et al., 2023; Stripelis et al., 2024; Ong et al., 2024; Zhuang et al., 2024; Feng et al., 2024; Lu et al., 2024; Zhao et al., 2024).

Existing works largely focus on routing over a *fixed* pool of LLMs, typically two. In practice, however, the pool of candidate LLMs can constantly change; e.g., older LLMs may be deprecated in favor of new, performant LLMs. Ideally, a router ought to leverage these new LLMs. To achieve this, perhaps the simplest approach is to retrain the router as the candidate pool varies. However, with frequent changes to the LLM pool, such retraining may be impractical owing to the non-trivial costs of both model retraining, as well as obtaining suitable *training labels* for each new LLM.

In this paper, we formalise this *dynamic* routing problem, wherein unobserved LLMs are available at test time. We propose an approach to this problem that relies on representing each LLM as a *feature vector*, derived based on *prediction correctness* on a set of representative prompts. Based on this, we detail an effective strategy that relies on cluster-based routing. Our solution allows an existing router to employ these test-time LLMs without retraining.

2 BACKGROUND

Language models (LMs). Given a finite vocabulary $\mathcal{V}$ of *tokens*, a *language model* (*LM*) is a distribution $p \in \Delta(\mathcal{V}^*)$, where $\mathcal{V}^* \doteq \bigcup_{n=0}^{\infty} \mathcal{V}^n$ and $\Delta(\cdot)$ denotes the set of distributions over a set. Often,

an LM is specified via *next-token probabilities* $\{p_{\text{NT}}(\cdot \mid \boldsymbol{x}) \in \Delta(\mathcal{V} \cup \{\perp\}) \colon \boldsymbol{x} \in \mathcal{V}^*\}$, for special terminal symbol $\perp \notin \mathcal{V}$. Concretely, for any $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{V}^n$, one constructs $p(y_1, \ldots, y_n) = p_{\text{NT}}(\perp \mid \boldsymbol{y}) \cdot \prod_{i=1}^{n} p_{\text{NT}}(y_i \mid \boldsymbol{y}_{<i})$ (Cotterell et al., 2024). Here, $\boldsymbol{y}_{<i} \doteq (y_1, \ldots, y_{i-1})$.

LLMs for predictive tasks. Let $\mathcal{X} \subset \mathcal{V}^*$ be a set of *input prompts*, and $\mathcal{Y}$ be a set of *targets*. We assume there is some underlying (unknown) distribution $\mathbb{P}$ over $\mathcal{X} \times \mathcal{Y}$. Let $\ell \colon \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ denote a *loss function*, where $\ell(\boldsymbol{x}, \boldsymbol{y}, \hat{\boldsymbol{y}})$ measures the *loss* (or *disutility*) of a *predicted* response $\hat{\boldsymbol{y}}$ on a given (prompt, target) pair $(\boldsymbol{x}, \boldsymbol{y})$. Our goal is to construct a *predictor* $h \colon \mathcal{X} \to \mathcal{Y}$ with low expected loss or *risk* $R(h) \doteq \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathbb{P}}\left[\ell(\boldsymbol{x}, \boldsymbol{y}, h(\boldsymbol{x}))\right]$.

An LLM natively provides a distribution over $\mathcal{V}^*$. To convert this to a predicted target, we assume there is some *prediction function* $\texttt{predict} \colon \Delta(\mathcal{V}^*) \to \mathcal{Y}$; e.g., if $\mathcal{Y} \subset \mathcal{V}^*$, we may employ a standard decoding algorithm (Ficler & Goldberg, 2017; Fan et al., 2018; Holtzman et al., 2020). Given such a function, we may construct $h(\boldsymbol{x}) \doteq \texttt{predict}(p(\cdot \mid \boldsymbol{x}))$, and seek to minimise $R(h)$.

Model routing. Model routing is a means for achieving efficiency at inference time by selecting an appropriate LLM for each input prompt. Inference efficiency is gained by sparingly calling a large model only on complex input prompts. Suppose we have a collection of $M \geq 2$ LLMs $\{h^{(m)}\}_{m\in[M]}$, with corresponding *inference costs* $\{c^{(m)}\}_{m\in[M]}$ denoting, e.g., the average latency of invoking each LLM. We assume $c^{(1)} \leq c^{(2)} \leq \ldots \leq c^{(M)}$. Let $r \colon \mathcal{X} \to [M]$ be a router that, given a prompt, predicts the most suitable LLM. We seek a router which achieves

$$\begin{aligned} &\min_{r \colon \mathcal{X}\to[M]} \sum_{m\in[M]} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\left[\mathbf{1}(r(\boldsymbol{x}) = m) \cdot \ell(\boldsymbol{x}, \boldsymbol{y}, h^{(m)})\right] : \\ &\text{subject to} \sum_{m\in[M]} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})}\left[\mathbf{1}(r(\boldsymbol{x}) = m) \cdot c^{(m)}\right] \leq B. \end{aligned} \tag{1}$$

Here, $B \in \mathbb{R}_+$ is some fixed budget on the cost of the routed solution.

Evaluation: deferral curve. We generally measure performance via a *deferral curve* (Jitkrittum et al., 2023; Wang et al., 2024a; Hu et al., 2024). This is a curve $\mathcal{C} = \{(B, R(h_{\text{RM}}(\cdot, r_B)) \colon B \in [c^{(1)}, c^{(M)}]\}$ tracing the tradeoff between the cost budget B and loss of the resulting routed model. Specifically, one varies the cost budget $B \in [c^{(1)}, c^{(M)}]$; computes a router $r_B(\cdot)$ for this budget; and plots the resulting expected loss $R(h_{\text{RM}}(\cdot, r_B))$. We may also use a quality metric (e.g., accuracy) in place of the loss to capture quality-cost trade-offs.

Model routing strategies. Narasimhan et al. (2022); Hu et al. (2024) proposed a post-hoc approach to model routing. Here, one constructs an estimator of the expected loss incurred by each LLM *in a fixed pool*, and route by picking the LLM with the lowest estimated loss, after appropriate cost adjustment. Different estimators have been proposed including a K-NN estimator (Hu et al., 2024), and a matrix factorisation approach (Ong et al., 2024).

3 Model Routing with a Dynamic LLM Pool

We now introduce the *dynamic model routing* problem. Suppose $\mathcal{H}_{\text{all}}$ denotes the set of all possible feasible LLM predictors, where we assume $|\mathcal{H}_{\text{all}}| < +\infty$. Let $\mathbb{H} \doteq 2^{\mathcal{H}_{\text{all}}}$ denote the set of all subsets of $\mathcal{H}$. Let $\mathcal{H}_{\text{tr}} = \{h^{(1)}, \ldots, h^{(M)}\} \in \mathbb{H}$ denote the set of M LLM predictors observed during training. During evaluation, we seek to route amongst the LLM predictors in some set $\mathcal{H}_{\text{te}} = \{h_{\text{te}}^{(1)}, \ldots, h_{\text{te}}^{(N)}\} \in \mathbb{H}^N$. If $\mathcal{H}_{\text{tr}} = \mathcal{H}_{\text{te}}$, we obtain the original model routing problem in (1). However, we aim to support settings where $\mathcal{H}_{\text{tr}} \neq \mathcal{H}_{\text{te}}$.

To accommodate the dynamic nature of evaluation LLMs, we first modify our router to accept both an input *and* a set of candidate LLMs, with the goal to pick the best option from this set; i.e., we consider *dynamic routers* $\mathcal{R} \doteq \{r(\cdot, \mathcal{H}) \colon \mathcal{X} \to [|\mathcal{H}|] \mid \mathcal{H} \in \mathbb{H}\}$. Next, we assume that the set of LLMs observed during training is itself drawn from some *meta-distribution* $\mathfrak{H}$ over $\mathbb{H}$. Rather than perform well on the specific set of training LLMs, we would like to generalise to *any* set of LLMs drawn from $\mathfrak{H}$. (This formulation is inspired by Tailor et al. (2024), developed in a related context.)

We now summarise the dynamic LLM routing problem as:

$$\min_{r \in \mathcal{R}} \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y},\mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x}, \mathcal{H}) = m) \cdot \ell(\boldsymbol{x}, \boldsymbol{y}, h^{(m)}) \right] :$$
$$\mathbb{E}_{(\boldsymbol{x},\boldsymbol{y},\mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x}, \mathcal{H}) = m) \cdot c(h^{(m)}) \right] \leq B, \tag{2}$$

where as before $B \in \mathbb{R}_+$ denotes a cost budget, $\mathcal{H} \doteq \{h^{(1)}, \ldots, h^{(M)}\} \sim \mathfrak{H}$ denotes a sample of M LLMs, and $c \colon \mathcal{H}_{\text{all}} \to \mathbb{R}_+$ denotes the cost of a given LLM.

3.1 Optimal Routing with a Dynamic Pool

To guide the design of a suitable dynamic router, we begin by studying the nature of the *Bayes-optimal* rule for (2). Note that the result closely mirrors Tailor et al. (2024, Eq. 6), and may be seen as a generalization of Lemma F.1 of Jitkrittum et al. (2023) to an arbitrary loss.

Proposition 1 (Optimal dynamic routing)**.** *Under a mild regularity condition on $\mathbb{P}$, for any input $\boldsymbol{x} \in \mathcal{X}$, LLM candidate set $\mathcal{H} \in \mathbb{H}$, and budget $B > 0$, the optimal dynamic router r^* for the constrained optimization in (2) is*

$$r^*(\boldsymbol{x}, \mathcal{H}) = \operatorname*{argmin}_{m \in [|\mathcal{H}|]} \left[\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}} \left[\ell(\boldsymbol{x}, \boldsymbol{y}, h^{(m)}) \right] + \lambda_{\mathfrak{H}} \cdot c(h^{(m)}) \right], \tag{3}$$

where $\lambda_{\mathfrak{H}} \geq 0$ is a Lagrange multiplier.

Intuitively, it is optimal to route to the model that has the lowest expected loss on the given input $\boldsymbol{x}$, after applying a *cost adjustment* of $\lambda_{\mathfrak{H}} \cdot c(h^{(m)})$ to the loss. The hyperparameter $\lambda_{\mathfrak{H}} \geq 0$ allows one to trade off the expected quality and the average cost.

3.2 Parameterising a Dynamic Router

A plug-in estimator to the optimal routing rule in (3) is constructed by substituting the per-example loss $\gamma(\boldsymbol{x}, h) \doteq \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}[\ell(\boldsymbol{x}, \boldsymbol{y}, h)]$ with an estimator $\hat{\gamma}(\boldsymbol{x}, h)$. To accommodate unseen LLMs, we construct a generic *LLM feature map* $\boldsymbol{\Psi} \colon \mathcal{H}_{\text{all}} \to \mathbb{R}^{D'}$. We may then compute $\hat{\gamma}(\boldsymbol{x}, h_{\text{te}}^{(n)}) = F(\boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Psi}(h_{\text{te}}^{(n)}))$, for suitable $F \colon \mathbb{R}^D \times \mathbb{R}^{D'} \to \mathbb{R}$, where $\boldsymbol{\Phi} \colon \mathcal{V}^* \to \mathbb{R}^D$ maps a query to a dense vector representation. Concretely, for any set of test LLMs $\mathcal{H}_{\text{te}} = \{h_{\text{te}}^{(n)}\}_{n \in [N]}$, we may estimate (3) via

$$\hat{r}(\boldsymbol{x}, \mathcal{H}_{\text{te}}) = \operatorname*{argmin}_{n \in [N]} \left[\hat{\gamma}(\boldsymbol{x}, h_{\text{te}}^{(n)}) + \lambda \cdot c(h_{\text{te}}^{(n)}) \right]. \tag{4}$$

Input prompt representation $\boldsymbol{\Phi}$. Following prior work (Hu et al., 2024), a natural choice for $\boldsymbol{\Phi}(\boldsymbol{x})$ is a frozen general-purpose text embedding, such as `text-embedding-3` (OpenAI, 2025), NV-Embed (Lee et al., 2025), E5-Mistral-7B (Wang et al., 2024b), and Gecko (Lee et al., 2024). An alternate approach is to construct a binary vector of *query attributes*, denoting whether a query possesses certain characteristics (Li et al., 2024a;b), e.g., whether it requires multi-step reasoning, seeks a single correct answer, and so on.

3.3 Correctness-Based LLM Representation

To construct our LLM representation $\boldsymbol{\Psi}$, it is useful to consider the properties a "good" representation ought to satisfy. One intuitive requirement is that $\boldsymbol{\Psi}(h)^\top \boldsymbol{\Psi}(h')$ should be large for a pair (h, h') of "similar" LLMs, and small for a pair of "dissimilar" LLMs. We posit that two LLMs are similar if they *have comparable performance on a set of representative prompts*, following similar proposals in Thrush et al. (2024); Zhuang et al. (2024).

Concretely, suppose that we have access to a small validation set $S_{\text{val}} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N_{\text{val}}}$ of labelled prompts. Further, suppose that *any new LLM $h_{\text{te}}^{(n)} \in \mathcal{H}_{\text{te}}$ can be evaluated on these prompts.*

We propose to represent any new LLM $h_{\text{te}}^{(n)}$ through its average errors $\hat{\boldsymbol{\Psi}}(h_{\text{te}}^{(n)}) \in [0,1]^K$ on K pre-defined *clusters*. We then approximate (3) via

$$\hat{\gamma}_{\text{clust}}(\boldsymbol{x}, h_{\text{te}}^{(n)}) \doteq \mathbf{z}(\boldsymbol{x})^\top \hat{\boldsymbol{\Psi}}(h_{\text{te}}^{(n)}).$$

where $\mathbf{z}(\boldsymbol{x}) \in \{0,1\}^K$ is a one-hot vector indicating the cluster membership of $\boldsymbol{x} \in \mathcal{V}^*$.

One challenge with the above is that clustering S_{val} itself is prone to overfitting, since (by assumption) the set is of modest size. To overcome this, we assume we have access to a large *unlabeled* training set consisting of input prompts $S_{\text{tr}} = \{\boldsymbol{x}^{(i)}\}_{i=1}^{N_{\text{tr}}}$. We now use the training set to group the prompts into K disjoint clusters using the K-means algorithm (MacQueen, 1967) (i.e., find K means in the embedding space provided by $\boldsymbol{\Phi}$), and compute per-cluster errors for a new LM using the validation set S_{val}.

Intuitively, $\hat{\gamma}_{\text{clust}}(\boldsymbol{x}, h_{\text{te}}^{(n)})$ estimates the performance of a given LLM on $\boldsymbol{x}$ by examining the performance of the LLM on *similar* prompts, i.e., those prompts belonging to the same cluster. Note that to add a new LM to the serving pool, we simply need to compute per-cluster errors $\hat{\boldsymbol{\Psi}}(h_{\text{te}}^{(n)})$ by generating responses from the LM on a small set of validation prompts. Importantly, this operation does not require any expensive gradient updates. Note that Li (2025) also considered learning model identity vectors to represent and accommodate new LLMs for model routing. In contrast to the approach of Li (2025) which is based on Item Response Theory and variational inference, we opt for a simple approach of correctness-based representation which does not require representation learning.

4 Experiments

We demonstrate the effectiveness of our proposed method on four benchmark datasets: for evaluating routing algorithms: EmbedLLM (Zhuang et al., 2024), MixInstruct (Jiang et al., 2023), RouterBench (Hu et al., 2024), and Chatbot Arena (Zheng et al., 2023).

Data pre-processing. With EmbedLLM, MixInstruct, and RouterBench, we partition the set of LLMs available into two disjoint sets: training models ($\mathcal{H}_{\text{tr}}$ in §3.3) and testing models ($\mathcal{H}_{\text{te}}$). For EmbedLLM which contains responses from as many as 112 LLMs, we use a random subset of $^2/_3$ for training and $^1/_3$ for testing. For MixInstruct (11 LLMs in total) and RouterBench (11 LLMs in total), we use a random 50% for training and the rest for testing. We randomly split examples into 60%/10%/30% for training, validation, and testing, respectively. All baselines are evaluated on the test portion and *only* on the test LLMs.

Per-example metrics. All the baselines we consider seek to estimate $\gamma^{(m)}(\boldsymbol{x}) = \mathbb{P}\left[\boldsymbol{y} \neq h^{(m)}(\boldsymbol{x}) \mid \boldsymbol{x}\right]$ and rely on the same deferral rule described in (3). For MixInstruct, we use (exponentiated) BARTScore (Yuan et al., 2021) as the evaluation metric, per Jiang et al. (2023).

Routing methods. For all the baselines that require a query embedder, we use the Gecko 1B model (Lee et al., 2024) to produce a 768-dimensional embedding. The Gecko checkpoint is used as is without further fine-tuning. The following are the routing methods we evaluate:

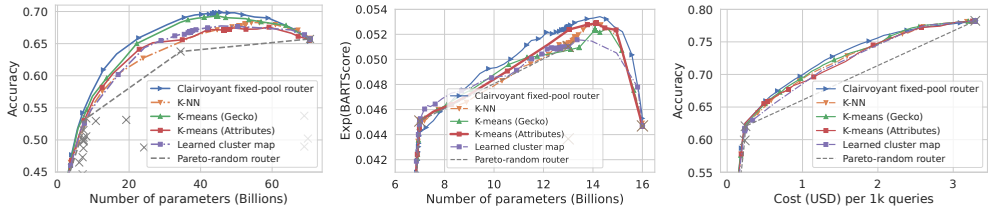
(1) **Clairvoyant fixed-pool router** (Hu et al., 2024; Ong et al., 2024; Ding et al., 2024). This is a representative baseline for the multi-model routing strategies described in §2 for a fixed (non-dynamic) pool of LLMs. Since these methods cannot route to unseen LLMs, we provide access to both the training and testing models during training (hence clairvoyant) to train a separate output head for each LLM to predict correctness labels. We use the sigmoid cross entropy loss on EmbedLLM and RouterBench, and the squared loss on MixInstruct. This baseline provides an estimate of the *performance achievable when all LLMs are observed.*

(2) **Pareto-random router.** For each prompt, the router routers to the LLM that achieves the highest cost-adjusted accuracy estimated in the validation set.

(3) **K-NN** (Hu et al., 2024). For each test prompt, this method looks up the K nearest prompts in the validation set in the space of Gecko embeddings and computes $\hat{\gamma}$ of each test LLM.

(4) **K-means (Gecko).** This is our proposed cluster-based routing rule in §3.

(5) **K-means (Attributes).** Same as the above proposed cluster-based routing, except that the query representation is based on the query attributes described in §3.2. We use the seven binary difficulty attributes proposed in Li et al. (2024a), and prompt Gemini 1.5 Pro 002 to annotate each attribute



(a) EmbedLLM (b) MixInstruct (c) RouterBench

Method \ Dataset	EmbedLLM			MixInstruct			RouterBench		
	Area ↑	QNC ↓	Peak Acc.↑	Area ↑	QNC ↓	Peak Acc.↑	Area ↑	QNC ↓	Peak Acc.↑
Pareto-random router	.611	1.0	65.7%	.0477	1.0	.0510	.692	1.0	78.2%
K-NN	.634	.51	68.3%	.0488	.94	.0528	.717	.962	78.3%
K-means (Gecko)	.650	.35	69.3%	.0489	.96	.0524	.720	.961	78.2%
K-means (Attributes)	.637	.50	67.6%	.0491	.90	.0529	.716	.970	78.2%
Clairvoyant fixed-pool router	.657	.32	69.9%	.0497	.85	.0534	.723	.989	78.2%

Figure 1: **Top**: Accuracy-cost trade-off curves (deferral curves) of the five methods (§4) on on unseen, testing LLMs. Our proposed routing approaches *K-means* are able to generalize to new LLMs, and deliver a good quality-cost trade-off. **Bottom**: Summary of each deferral curve. On all datasets, our proposal yields the highest area under the curve, and lowest Quality-Neutral Cost (QNC) i.e., the minimum relative cost to achieve the same performance as the most accurate LLM. Pareto-random router and K-NN are the main baselines. The Clairvoyant fixed-pool router represents an upper bound on the performance achievable had *all* LLMs been observed during training.

for each training prompt. We then construct a query embedder $\Phi(\boldsymbol{x}) = \sigma(\mathbf{V}^\top \text{Gecko}(\boldsymbol{x})) \in [0,1]^7$, where $\mathbf{V} \in \mathbb{R}^{768\times 7}$ is distilled using the training set to predict the 7-category attributes for any new prompt $\boldsymbol{x}$. This compact query representation can be robust to shifts in the query distribution at test time (see §C.3 for additional experiments).

Deferral curve. We evaluate each method with a deferral curve as described in §2, which plots the average quality against the overall cost. The trade-off is realized by varying the $\lambda_{\mathfrak{H}}$ parameter in the routing rule in (3). For EmbedLLM and MixInstruct, we use the number of parameters of the LLM as the cost of processing one prompt. This cost definition is a convenient proxy for the amount of computation required to call each LLM. For RouterBench, we plot LLMs' API calling costs (USD) as available in the dataset.

Hyper-parameter tuning. For each K (number of neighbors in K-NN and the number of clusters in our proposal), we represent the training LLM using correctness labels in the training set, evaluate the routing rule for the training LLMs on the validation set, and measure the area under the deferral curve. The parameter K with the maximum area is then chosen. See §C.1 for details.

Results. We present deferral curves for different methods in Figure 1. Each isolated point x represents the cost and average test accuracy of one testing LLM. In the table, we report three evaluation metrics for each method: (i) the area under the deferral curve (Area); (ii) the quality-neutral cost (QNC) or the minimum relative cost needed to achieve the same performance as the most accurate testing LLM; and (iii) the peak accuracy (Peak Acc.) achievable across the entire cost range.

Our proposed method *K-means (Gecko)* yields the best quality-cost trade-off in most cases. Interestingly, the *K-means (Attributes)*, which relies on only a 7-dimensional input representation, performs remarkably well on MixInstruct. Notably, on EmbedLLM, our K-means (Gecko) almost matches the performance of Clairvoyant fixed-pool router, *despite not observing testing models during training*. An important comparison point is the Pareto-random router, which was noted as a strong baseline in Hu et al. (2024) (referred to as the Zero Router). On both EmbedLLM and RouterBench, our methods are able to exceed the performance of this baseline on a large range of costs.

The surprising underperformance of K-NN is due, in large part, to the requirement that only the retrieved neighbors from the validation set can be used to estimate test models' performance. It is hence unable to exploit the training set in any way. In contrast, our methods are able to fully exploit the training data either in an unsupervised manner (K-means).

5 CONCLUSION

We present principled strategies for routing amongst multiple unseen test-time LLMs, without having to retrain the router. Central to our approach is a *prediction correctness* LLM representation, with accompanying cluster-based routing strategies. Our proposal is computationally efficient, and able to deliver a good quality-cost trade-off as shown in experiments involving > 30 unseen LLMs on EmbedLLM. An interesting future direction is to enhance routing robustness to query distribution shifts. Such a routing system will further reduce the need for frequent router retraining.

REFERENCES

Peter L. Bartlett and Marten H. Wegkamp. Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840, 2008. URL http://jmlr.org/papers/v9/bartlett08a.html.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. In *International Conference on Machine Learning*, 2017.

Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *International Conference on Machine Learning (ICML)*, July 2024a.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024b.

Boyuan Chen, Mingzhi Zhu, Brendan Dolan-Gavitt, Muhammad Shafique, and Siddharth Garg. Model cascading for code: Reducing inference costs with model cascading for llm based code generation, 2024a. URL https://arxiv.org/abs/2405.15842.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023b.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=7RQvjayHrM.

C Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *ALT*, 2016. URL https://cs.nyu.edu/~mohri/pub/rej.pdf.

Ryan Cotterell, Anej Svete, Clara Meister, Tianyu Liu, and Li Du. Formal aspects of language modeling, 2024. URL https://arxiv.org/abs/2311.04329.

Fnu Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit S Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham M. Kakade, Ali Farhadi, and Prateek Jain. Matformer: Nested transformer for elastic inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=fYa6ezMxD5.

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.

Arthur Douillard, Qixuan Feng, Andrei A. Rusu, Adhiguna Kuncoro, Yani Donchev, Rachita Chhaparia, Ionel Gog, Marc'Aurelio Ranzato, Jiajun Shen, and Arthur Szlam. Dipaco: Distributed path composition, 2024. URL https://arxiv.org/abs/2403.10616.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.681. URL http://dx.doi.org/10.18653/v1/2024.acl-long.681.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A graph-based router for llm selections, 2024. URL https://arxiv.org/abs/2410.03834.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pp. 94–104, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL https://aclanthology.org/W17-4912.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. 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. 2151–2159. PMLR, 09–15 Jun 2019.

Neel Guha, Mayee F Chen, Trevor Chow, Ishan S. Khare, and Christopher Re. Smoothie: Label free language model routing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=pPSWHsgqRp.

Neel Guha, Mayee F Chen, Trevor Chow, Ishan S. Khare, and Christopher Re. Smoothie: Label free language model routing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=pPSWHsgqRp.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KgaBScZ4VI.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models, 2023. URL https://arxiv.org/abs/2308.11601.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. How good are GPT models at machine translation? a comprehensive evaluation, 2023. URL https://arxiv.org/abs/2302.09210.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH.

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.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Exploring the benefits of training expert language models over instruction tuning. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Wittawat Jitkrittum, Neha Gupta, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=4KZhZJSPYU.

M.I. Jordan and R.A. Jacobs. Hierarchical mixtures of experts and the em algorithm. In *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, volume 2, pp. 1339–1344 vol.2, 1993. doi: 10.1109/IJCNN.1993.716791.

Anil Kag, Igor Fedorov, Aditya Gangrade, Paul Whatmough, and Venkatesh Saligrama. Efficient edge inference by selective query. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jpR98ZdIm2q.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2025. URL https://arxiv.org/abs/2405.17428.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. Gecko: Versatile text embeddings distilled from large language models, 2024. URL https://arxiv.org/abs/2403.20327.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Tianle Li, Wei-Lin Chiang, and Lisa Dunlap. Introducing hard prompts category in chatbot arena. https://lmsys.org/blog/2024-05-17-category-hard, 2024a.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024b. URL https://arxiv.org/abs/2406.11939.

Yang Li. Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing, 2025. URL https://arxiv.org/abs/2502.02743.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024c.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 7421–7432, Miami, Florida, USA, November 2024d. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.422. URL https://aclanthology.org/2024.emnlp-main.422/.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human*

Language Technologies (Volume 1: Long Papers), pp. 1964–1974, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.109. URL https://aclanthology.org/2024.naacl-long.109/.

J. MacQueen. Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NeurIPS'18, pp. 6150–6160, Red Hook, NY, USA, 2018. Curran Associates Inc.

Anqi Mao, Christopher Mohri, Mehryar Mohri, and Yutao Zhong. Two-stage learning to defer with multiple experts. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024a. Curran Associates Inc.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Realizable h-consistent and Bayes-consistent loss functions for learning to defer. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=OcO2XakUUK.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Regression with multi-expert deferral. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 34738–34759. PMLR, 21–27 Jul 2024c. URL https://proceedings.mlr.press/v235/mao24d.html.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7076–7087. PMLR, 13–18 Jul 2020.

Harikrishna Narasimhan, Wittawat Jitkrittum, Aditya Krishna Menon, Ankit Singh Rawat, and Sanjiv Kumar. Post-hoc estimators for learning to defer to an expert. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=˙jg6Sf6tuF7.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding, 2024a. URL https://arxiv.org/abs/2405.19261.

Harikrishna Narasimhan, Aditya Krishna Menon, Wittawat Jitkrittum, Neha Gupta, and Sanjiv Kumar. Learning to reject for balanced error and beyond. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=ta26LtNq2r.

Harikrishna Narasimhan, Aditya Krishna Menon, Wittawat Jitkrittum, and Sanjiv Kumar. Plugin estimators for selective classification with out-of-distribution detection. In *The Twelfth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=DASh78rJ7g.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data, 2024. URL https://arxiv.org/abs/2406.18665.

OpenAI. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2025.

Mathieu Ravaut, Shafiq Joty, and Nancy F Chen. Summareranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. *arXiv preprint arXiv:2203.06569*, 2022.

Ankit Singh Rawat, Manzil Zaheer, Aditya Krishna Menon, Amr Ahmed, and Sanjiv Kumar. When in doubt, summon the titans: Efficient inference with large models. *CoRR*, abs/2110.10305, 2021a.

Ankit Singh Rawat, Manzil Zaheer, Aditya Krishna Menon, Amr Ahmed, and Sanjiv Kumar. When in doubt, summon the titans: Efficient inference with large models. *arXiv preprint arXiv:2110.10305*, 2021b.

Sara Sangalli, Ertunc Erdil, and Ender Konukoglu. Expert load matters: operating networks at high accuracy and low manual effort. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Y2VQWfi7Vc.

Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks? *Cognitive Computation*, 12:954–966, 2020.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=uLYc4L3C81A.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Sidak Pal Singh and Martin Jaggi. Model fusion via optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22045–22055. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb2697869f56484404c8ceee2985b01d-Paper.pdf.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *CoRR*, abs/1811.03115, 2018. URL http://arxiv.org/abs/1811.03115.

Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Zhaozhuo Xu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Salman Avestimehr, and Chaoyang He. Tensoropera router: A multi-model router for efficient llm inference, 2024. URL https://arxiv.org/abs/2408.12320.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. Multi-mention learning for reading comprehension with neural cascades. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HyRnez-RW.

Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. Learning to Defer to a Population: A Meta-Learning Approach. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*, 2024.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, pp. 2464–2469. IEEE, 2016.

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations, 2024. URL https://arxiv.org/abs/2409.05816.

Vivien Tran-Thien. An optimal lossy variant of speculative decoding, 2023. URL https://github.com/vivien000/mentored_decodings. Unsupervised Thoughts (Blog). URL: https://github.com/vivien000/mentored_decoding.

Kirill Trapeznikov and Venkatesh Saligrama. Supervised sequential classification under budget constraints. In Carlos M. Carvalho and Pradeep Ravikumar (eds.), *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pp. 581–589, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pp. I–I, 2001. doi: 10.1109/CVPR.2001.990517.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? Cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, WSDM '24, pp. 606–615, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400703713. doi: 10.1145/3616855.3635825. URL https://doi.org/10.1145/3616855.3635825.

Congchao Wang, Sean Augenstein, Keith Rush, Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Aditya Krishna Menon, and Alec Go. Cascade-aware training of language models, 2024a. URL https://arxiv.org/abs/2406.00060.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024b. URL https://arxiv.org/abs/2401.00368.

Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M. Kitani, Yair Movshovitz-Attias, and Elad Eban. Wisdom of committees: An overlooked approach to faster and more accurate models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=MvO2t0vbs4-.

Xin Wang, Yujia Luo, Daniel Crankshaw, Alexey Tumanov, Fisher Yu, and Joseph E. Gonzalez. IDK cascades: Fast deep learning by learning not to overthink. In Amir Globerson and Ricardo Silva (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 580–590. AUAI Press, 2018.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. Early exiting BERT for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pp. 83–88, Online, November 2020. Association for Computational Linguistics.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. BARTSCORE: evaluating generated text as text generation. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thought representations for cost-efficient reasoning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6okaSfANzh.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4447–4462, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.263. URL https://aclanthology.org/2024.findings-acl.263/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=uccHPGDlao.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y Zhao, Andrew M. Dai, Zhifeng Chen, Quoc V Le, and James Laudon. Mixture-of-experts with expert choice routing. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=jdJo1HIVinI.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models, 2024. URL https://arxiv.org/abs/2410.02223.

Universal LLM Routing with Correctness-Based Representation

Supplementary Material

A Proof of Proposition 1

In what follows, we use $r_{\mathcal{H}}(\boldsymbol{x})$ and $r(\boldsymbol{x}, \mathcal{H})$ interchangeably.

Proof. The constrained problem in (2) is equivalent to minimizing the following Lagrangian objective for some Lagrange multiplier $\lambda_{\mathfrak{H}} \geq 0$:

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y},\mathcal{H})}\left[\sum_{m\in[|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x},\mathcal{H})=m)\cdot \ell(\boldsymbol{x},\boldsymbol{y},h^{(m)})\right] + \lambda_{\mathfrak{H}}\cdot \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y},\mathcal{H})}\left[\sum_{m\in[|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x},\mathcal{H})=m)\cdot c^{(m)}\right] \\
&\overset{(a)}{=} \mathbb{E}_{\mathcal{H}}\mathbb{E}_{\boldsymbol{x}}\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\left[\sum_{m\in[|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x},\mathcal{H})=m)\cdot \left\{\ell(\boldsymbol{x},\boldsymbol{y},h^{(m)}) + \lambda_{\mathfrak{H}}\cdot c^{(m)}\right\}\right] \\
&= \mathbb{E}_{\mathcal{H}}\mathbb{E}_{\boldsymbol{x}}\Bigg[\underbrace{\sum_{m\in[|\mathcal{H}|]} \mathbf{1}(r(\boldsymbol{x},\mathcal{H})=m)\cdot \left\{\mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\left[\ell(\boldsymbol{x},\boldsymbol{y},h^{(m)})\right] + \lambda_{\mathfrak{H}}\cdot c^{(m)}\right\}}_{\mathcal{L}_{\mathcal{H},\boldsymbol{x}}}\Bigg],
\end{aligned}$$

where (a) uses the fact that the draw of $\mathcal{H}$ is independent of the draw of $(\boldsymbol{x}, \boldsymbol{y})$. The last line makes it clear that for any fixed $\mathcal{H}$ and any fixed $\boldsymbol{x}$, to minimize the overall loss, the router ought to route to the model that has the lowest cost-adjusted loss $\mathcal{L}_{\mathcal{H},\boldsymbol{x}}$. Thus,

$$r^*(\boldsymbol{x},\mathcal{H}) = \text{argmin}_{m\in[|\mathcal{H}|]}\, \mathbb{E}_{\boldsymbol{y}|\boldsymbol{x}}\left[\ell(\boldsymbol{x},\boldsymbol{y},h^{(m)})\right] + \lambda_{\mathfrak{H}}\cdot c^{(m)}.$$

□

B Experimental Setup

We provide more details on the experiments discussed in Section 4.

B.1 Splitting Data and LLMs

In the experiment on each of the three datasets (EmbedLLM (Zhuang et al., 2024), MixInstruct (Jiang et al., 2023), and RouterBench (Hu et al., 2024)), we split the data into three disjoint portions: train (60%), validation (10%), and test (30%). The set of all LLMs available in each dataset is also split into two disjoint sets: training models and testing models. The relationship of data splits and model splits is summarized in the following table.

	Train (60%)	Validation (10%)	Test (30%)
Training models	✓	✓	✗
Testing models	✗	✓	✓

- **Training set**. The training examples are meant for router training. Only information of the training models (not testing models) is available in this data portion. The only exception is the clairvoyant fixed-pool router baseline which is allowed access to correctness labels of testing models on training examples. In other words, unlike other baselines, the clairvoyant fixed-pool router observes all models during training, and is trained on both training and validation portions. This baseline is meant to establish performance achievable if a router has access to all models.

- **Validation set**. The validation examples are meant to be used to represent new LLMs. For instance, for our proposed K-means (Gecko) approach, the validation set is used to compute per-cluster performance metrics of each testing LLM observed at test time, to represent it as a feature vector.
- **Test set**. The test examples are only used for evaluating routing methods by evaluating their deferral curves.

Testing models represent new models that arrive at deployment time, and are not available for training (except to the clairvoyant fixed-pool router baseline). Training models are meant for router training. For instance, our proposed K-means (Gecko) approach learns to route among the training models, and is tested on the test set to route among the testing models.

C Additional Experimental Results

We present additional experimental results we omitted in the main text.

C.1 Selecting K

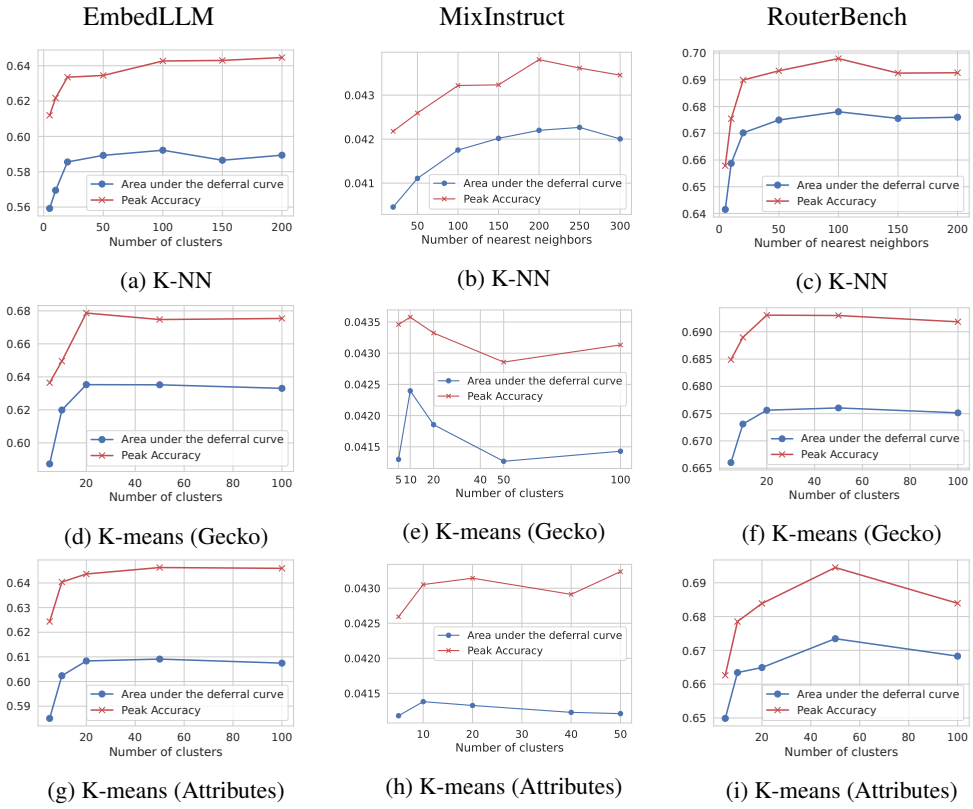
There are four baselines that we consider in the experiments in Section 4 that depend on a hyperparameter K. Specifically, K in K-NN refers to the number of nearest neighbors, and K in K-means (Gecko), and K-means (Attributes) refers to the number of clusters. In Figure 1, we report the performance of these methods with the best K found on each dataset separately for each method. We now describe the validation procedure we used to select the best K.

K-NN For each candidate K, and each query in the validation set, find the K nearest queries in the training set (in the Gecko embedding space). Route each query in the validation set to the most appropriate *training* LLM according to the routing rule (4). Produce a deferral curve on the validation set, and compute the normalized area under such curve. Select K that maximizes the area.

K-means (Gecko) For each candidate K, perform K-means on the training set using Gecko embeddings (Lee et al., 2024). Compute the feature vector representation of each *training* LLM on the training set as described in §3.3. For each query in the validation set, find the nearest cluster, and route the query to the most appropriate *training* LLM according to the routing rule (4). Produce a deferral curve on the validation set, and compute the normalized area under such curve. Select K that maximizes the area.

K-means (Attributes) Parameterize the query embedding model to be $\Phi(\boldsymbol{x}) = \sigma(\mathbf{V}^\top \mathrm{Gecko}(\boldsymbol{x}))$ where $\mathbf{V} \in \mathbb{R}^{768\times 7}$, and σ denotes the sigmoid function. Train each head $\mathbf{v}_j \in \mathbb{R}^{768}$ (with Gecko frozen) by minimizing the sigmoid cross entropy to predict whether the j-th semantic attribute is active on each input query. We use the seven prompt difficulty attributes as described in Li et al. (2024a), and prompt Gemini 1.5 Pro 002 to annotate each binary attribute on each training example. Once the query embedding model Φ is trained, we freeze it, and perform the same hyperparameter selection procedure as used for K-means (Gecko) by replacing the Gecko embedding function with Φ.

Figure 2 shows the area under the deferral curve (on the validation set) vs candidate parameter K. Importantly, the testing models and the test set are never used in the above hyperparameter selection process.



(a) K-NN (b) K-NN (c) K-NN

(d) K-means (Gecko) (e) K-means (Gecko) (f) K-means (Gecko)

(g) K-means (Attributes) (h) K-means (Attributes) (i) K-means (Attributes)

Figure 2: Validation performance of the three methods considered in Figure 1: K-NN, K-means (Gecko), and K-means (Attributes). See Appendix C.1 for more details.

C.2 Results on Individual Datasets in EmbedLLM

To supplement results on EmbedLLM in Figure 1a, we further evaluate the same router models separately on the 10 datasets contained in EmbedLLM. The results are shown in Figure 3.

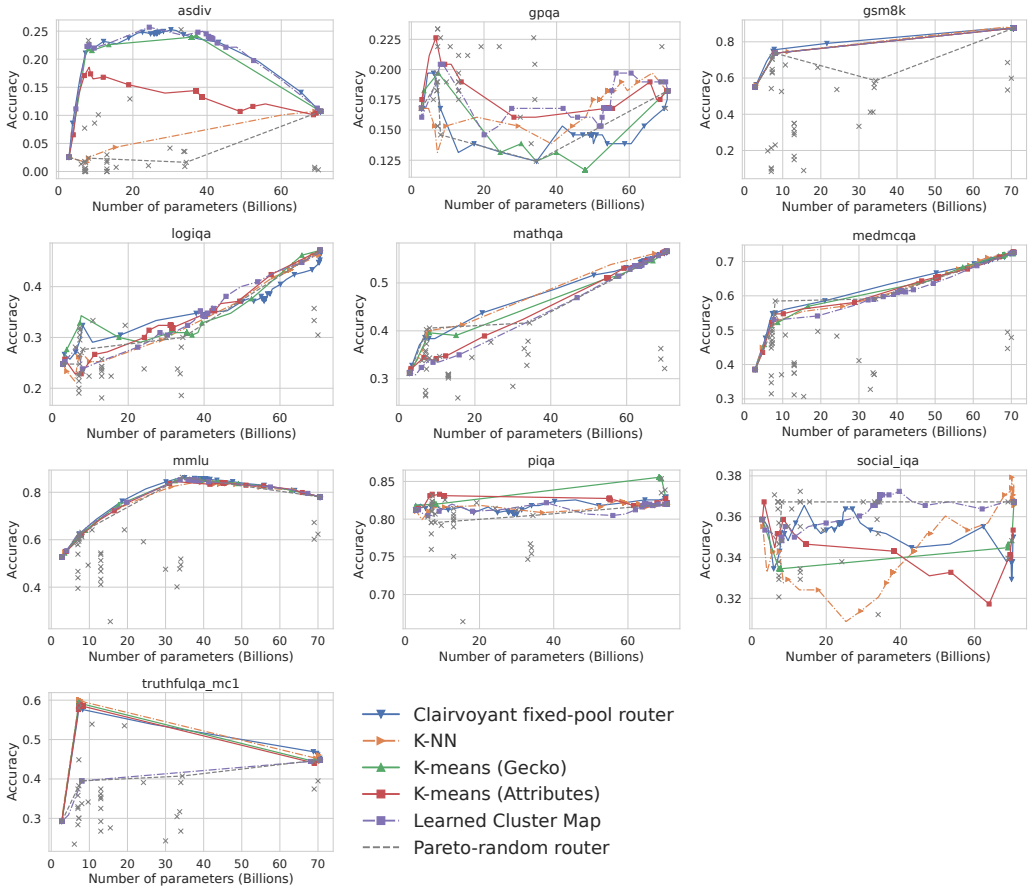
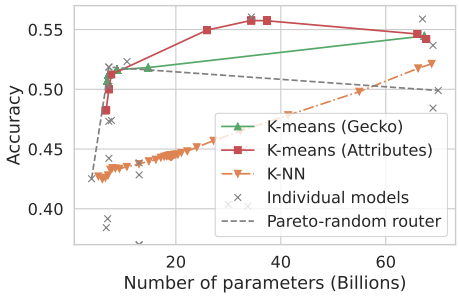


Figure 3: Evaluation of the router models used in Figure 1a on the 10 datasets contained in EmbedLLM.

C.3 Train on Chatbot Arena and Test on EmbedLLM



Method	Area↑	QNC↓	Peak Acc.↑
Pareto-random router	.507	∞	51.9%
K-NN	.472	∞	52.1%
K-means (Gecko)	.529	∞	54.5%
K-means (Attributes)	.545	.97	55.8%

Figure 4: Deferral curves of routers trained on Chatbot Arena with pairwise comparison labels, and tested on EmbedLLM with per-query correctness labels.

Representing each prompt with a small number of attributes that capture its inherent hardness shines when there is a query distribution shift at test time. To illustrate this, we compare Gecko-based query representation and attribute-based representation by training on Chatbot Arena conversation data (Zheng et al., 2023), and testing on EmbedLLM, which contains mostly Q&A prompts. To reduce confounding factors, we train on all LLMs that are present in both datasets (26 LLMs), and test on the same set of LLMs (i.e., no unseen LLMs at test time). After appropriate filtering, the Chatbot Arena dataset has 8447 records left. The filtering step ensures that we only deal with LLMs that are present in both datasets. These examples are split further into 90% training and 10% validation splits.

The Chatbot Arena dataset contains pairwise comparison labels: each user query is responded to by two random LLMs, to which the user selects the better response. To evaluate per-cluster performance for representing each LLM, we fit the Bradley-Terry-Luce model (Bradley & Terry, 1952) to the pairwise comparison labels within a cluster and estimate the pointwise quality scores for each LLM for that cluster. We use the full EmbedLLM dataset for testing.

The results are shown in Figure 4. We observe that K-means (Attributes) in this case performs better than K-means (Gecko), suggesting that using prompt hardness attributes helps improve robustness to query distribution shifts. In fact, this routing approach is the only method that can reach the performance of the most accurate model in the pool, thus attaining a finite quality-neutral cost (QNC). The reason the Pareto-random router has a decreasing trend is because the Pareto-optimal LLMs are chosen using the validation set, and turn out to be not optimal for the test set.

D Related work

Model routing. Model routing has emerged as a simple yet effective strategy to lower LLMs' inference cost (Hendy et al., 2023; Hari & Thomson, 2023). Recent works have studied various strategies to learn a router, including training an explicit "meta-model" based on a neuronal network (Ding et al., 2024; Šakota et al., 2024; Chen et al., 2024b), k-nearest neighbours (Hu et al., 2024; Shnitzer et al., 2023; Stripelis et al., 2024), matrix factorisation (Ong et al., 2024; Zhuang et al., 2024), and graph neural networks (Feng et al., 2024). Works have also explored the role of supervision in training a router (Lu et al., 2024; Zhao et al., 2024). Typically, the router orchestrates amongst multiple independent LLMs, although it is also possible to route amongst *implicit* sub-models in a larger model, such as those defined by a MatFormer (Devvrit et al., 2024; Cai et al., 2024a).

Model fusion Model routing may be contrast to model *fusion*, where the primary goal is to leverage multiple models to improve *quality*, potentially at the expense of *efficiency*. This can involve invoking multiple models prior to generating an output (Ravaut et al., 2022; Jiang et al., 2023; Guha et al., 2024b; Wang et al., 2024b), or producing a single fused model (Singh & Jaggi, 2020).

Mixture of experts (MoE). Classically, MoE models focused on learning parameters for independent models, along with a suitable routing rule (Jacobs et al., 1991; Jordan & Jacobs, 1993). These have proven an plausible alternative to model specialisation (Jang et al., 2023; Douillard et al., 2024). Such models are typically of the same size; thus, cost considerations do not factor into the router

design. More recently, MoEs have focussed on *sub*-models within a single larger model, e.g., a Transformer (Fedus et al., 2022; Zhou et al., 2022).

Model cascading. Cascading is a closely related technique for orchestrating amongst multiple models, wherein one *sequentially invokes* each model in order of cost. One then uses statistics from the resulting model output (e.g., the confidence) to decide whether or not to proceed to the next costlier model. Cascading has a long history in computer vision applications (Viola & Jones, 2001; Wang et al., 2018; Swayamdipta et al., 2018; Rawat et al., 2021b; Wang et al., 2022; Kag et al., 2023; Jitkrittum et al., 2023). Recently, cascades have also been successfully proven in the case of LLMs (Chen et al., 2023b; Gupta et al., 2024; Yue et al., 2024; Chen et al., 2024a).

Selective classification and learning to defer. The formal underpinnings of routing and cascading can be traced to three closely related literatures: learning to reject (Chow, 1970; Bartlett & Wegkamp, 2008; Cortes et al., 2016), selective classification (Geifman & El-Yaniv, 2019; Narasimhan et al., 2024b;c), and learning to defer to an expert (Madras et al., 2018; Sangalli et al., 2023). Following pioneering studies of Trapeznikov & Saligrama (2013); Bolukbasi et al. (2017); Mozannar & Sontag (2020), a series of works have studied the routing and cascading problem through these lenses (Narasimhan et al., 2022; Mao et al., 2024b;a;c).

Early-exiting. Early-exiting enables adaptive computation within a *single* neural model, by allowing computation to terminate at some intermediate layer (Teerapittayanon et al., 2016; Scardapane et al., 2020). Often, the termination decision is based on a simple model confidence (akin to simple model cascading), but learning approaches have also been considered (Xin et al., 2020; Schuster et al., 2022).

Speculative decoding. Speculative decoding is another technique that leverages two models to speed up inference, by using the smaller model to draft tokens and having the larger model verify them (Stern et al., 2018; Chen et al., 2023a; Leviathan et al., 2023; Tran-Thien, 2023; Sun et al., 2024; Zhou et al., 2024; Cai et al., 2024b; Li et al., 2024c;d). Recent works have studied approaches to combine speculative decoding with early-exits (Elhoushi et al., 2024) and cascades (Narasimhan et al., 2024a).

Routing approach	Candidate LLMs	Training signals	Works without task labels	Adaptive computation	Unseen models at test time	Reference
Smoothie	Any	Query embeddings. No label required.	✓	✘	✘	Guha et al. (2024a)
Cascading with token-level features	2	Pointwise evaluation.	✓	✓	✘	Gupta et al. (2024)
Summon the titans	2	Annotations from a teacher model.	✓	✓	✘	Rawat et al. (2021a)
RouteLLM	2	Pairwise comparison metrics.	✓	✓	✘	Ong et al. (2024)
K-NN router	Any	Pointwise evaluation, query embeddings.	✓	✓	✘	Hu et al. (2024)
GraphRouter	Any	Task information	✘	✓	✓	Feng et al. (2024)
Our proposal	Any	Pointwise evaluation, query clusters	✓	✓	✓	This work

Table 1: A qualitative comparison of recently proposed query routing approaches. Adaptive computation refers to the ability to trade quality for a reduced inference cost.