

Universal Model Routing for Efficient LLM Inference

Wittawat Jitkrittum
wittawat@google.com

Harikrishna Narasimhan
hnarasimhan@google.com

Ankit Singh Rawat
ankitsrawat@google.com

Jeevesh Juneja
jeeveshjuneja@google.com

Congchao Wang
congchaowang@google.com

Zifeng Wang
zifengw@google.com

Alec Go
ago@google.com

Chen-Yu Lee
chenyulee@google.com

Pradeep Shenoy
shenoypradeep@google.com

Rina Panigrahy
rinap@google.com

Aditya Krishna Menon
adityakmenon@google.com

Sanjiv Kumar
sanjivk@google.com

Google

Abstract

Model routing is a simple technique for reducing the inference cost of large language models (LLMs), 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 UniRoute, 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 two effective instantiations of UniRoute, relying on *cluster-based* routing and a *learned cluster map* respectively. We show that these are estimates of a theoretically optimal routing rule, and quantify their errors via an excess risk bound. Experiments on a range of public benchmarks show the effectiveness of UniRoute in routing amongst more than 30 unseen LLMs.

1 Introduction

Large language models (LLMs) have seen a flurry of recent development [Radford et al., 2018, 2019, Brown et al., 2020, Touvron et al., 2023, Anil et al., 2023, Grattafiori et al., 2024, DeepSeek-AI et al., 2024]. These impressive abilities notwithstanding, the inference cost of LLMs can be prohibitive [Li et al., 2024a, Wan et al., 2024, Zhou et al., 2024b]. This has motivated several techniques to improve LLM inference efficiency, such as speculative decoding [Stern et al., 2018, Chen et al., 2023a, Leviathan et al., 2023], early-exiting [Schuster et al., 2022], quantisation [Chee et al., 2023], compression [Frantar and Alistarh, 2023, Agarwal et al., 2024, Rawat et al., 2024], and others [Pope et al., 2023, S et al., 2024, Menghani, 2023].

Our interest is in *model routing* for efficient inference. Here, one maintains a pool of candidate LLMs of various sizes and capabilities. Given a prompt, one learns to predict the lowest-cost LLM which can reasonably address the prompt. 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 effective technique, which has seen a surge of recent interest [Hendy et al., 2023, Hari and Thomson, 2023, Ding et al., 2024, Šakota et al., 2024, Chen et al., 2024b, Hu et al., 2024b, Shnitzer et al., 2023, Wang et al., 2023, Stripelis et al., 2024, Ong et al., 2025, Zhuang et al., 2024, Srivatsa et al., 2024, Feng et al., 2024, Lu et al., 2024, Zhao et al., 2024b, Dann et al., 2024, Aggarwal et al., 2024, Lee et al., 2024a, Mohammadshahi et al., 2024, Chuang et al., 2025, Huang et al., 2025].

Existing works largely focus on routing over a *fixed* pool of LLMs. In practice, however, the pool of candidate LLMs can constantly change; e.g., older LLMs may be deprecated in favor of new, performant LLMs. To leverage such new LLMs, perhaps the simplest approach is to re-train the router. However, with frequent

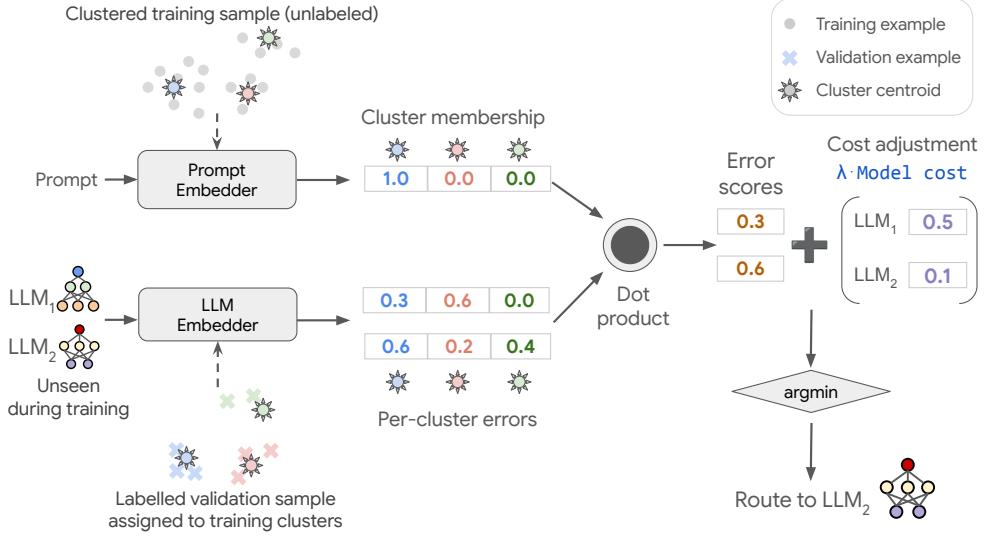


Figure 1: Illustration of our proposed UniRoute with a cluster-based router (see §5.1). We first perform K -means on a training set to find K centroids, and then partition the validation set into K representative clusters. Each test-time LLM can then be represented as a K -dimensional feature vector of per-cluster errors. This yields an intuitive routing rule: for each test prompt, we route to the LLM with the smallest cost-adjusted average error on the cluster the prompt belongs to. The prompt embedder may either be completely *unsupervised* (as shown in the figure), or fitted via *supervised* learning using labels from a set of training LLMs different from those seen during test time (§5.2).

changes to the LLM pool, this may be impractical owing to the non-trivial overhead of both model re-training, as well as obtaining sufficient *training labels* for each new LLM.

In this paper, we propose UniRoute, a new approach to this problem based on representing each LLM as a *feature vector*, derived from their *prediction errors* on a set of representative prompts. By learning a router over these LLM features, we enable generalisation to previously unseen LLMs *without* any re-training. Building on the observation that K -NN routing [Hu et al., 2024b] is a special case of UniRoute, we detail two concrete instantiations of UniRoute relying on *unsupervised* and *supervised* prompt clustering respectively. While conceptually simple, empirically, these enable effective routing with a dynamic LLM pool across several benchmarks. In sum, our contributions are:

- (i) we formalise the problem setting of model routing with a *dynamic* pool of LLMs (§3);
- (ii) we propose UniRoute (§4.1), a novel routing approach relying on an LLM feature representation based on its *prediction error vector* on a small set of representative prompts (§4.2);
- (iii) we propose two simple yet effective instantiations of UniRoute based on unsupervised or supervised clustering (§5.1, §5.2, Figure 1), with an accompanying excess risk bound (§5.3);
- (iv) we present experiments (§7) on EmbedLLM [Zhuang et al., 2024], SPROUT o3-mini [Somerstep et al., 2025], RouterBench [Hu et al., 2024b], and Chatbot Arena [Ong et al., 2025], illustrating the ability to effectively route amongst > 30 unseen LLMs.

2 Background: Model Routing with a Static LLM Pool

Language models (LMs). Given a finite, non-empty vocabulary of *tokens* \mathcal{V} , 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. *Large language models (LLMs)* based on Transformers [Vaswani et al., 2017] have proven highly versatile.

LLMs for predictive tasks. Our interest is in using LLMs for the following prediction problem. Let $\mathcal{X} \subset \mathcal{V}^*$ be a set of *input prompts*, and \mathcal{Y} be a set of *targets*. Let $\ell(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}})$ denote a *loss function* measuring the *loss* (or *disutility*) of a *predicted response* $\hat{\mathbf{y}}$ on a given (prompt, target) pair (\mathbf{x}, \mathbf{y}) . For example, $\ell(\mathbf{x}, \mathbf{y}, \hat{\mathbf{y}}) = \mathbf{1}(\mathbf{y} \neq \hat{\mathbf{y}})$ measures whether the response is an exact string match of the target. Our goal is to construct a *predictor* $h: \mathcal{X} \rightarrow \mathcal{Y}$ minimising $R(h) \doteq \mathbb{E}_{(\mathbf{x}, \mathbf{y}) \sim \mathbb{P}} [\ell(\mathbf{x}, \mathbf{y}, h)]$.

An LLM natively provides a distribution over \mathcal{V}^* . To convert such a distribution to a predicted target, we assume there is some (task-specific, and possibly randomised) *prediction function* $\text{predict}: \Delta(\mathcal{V}^*) \rightarrow \mathcal{Y}$; e.g., in the simple case where $\mathcal{Y} \subset \mathcal{V}^*$, we may employ a standard decoding algorithm [Ficler and Goldberg, 2017, Fan et al., 2018, Holtzman et al., 2020]. More generally, predict may involve some non-trivial processing (e.g., stripping non-numeric tokens). Given such a function, we may construct $h(\mathbf{x}) \doteq \text{predict}(p(\cdot | \mathbf{x}))$ 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 “hard” input prompts. More precisely, suppose we have a set of $M \geq 2$ LLMs $\{p^{(m)}\}_{m \in [M]}$, with corresponding *inference costs* $\{c^{(m)}\}_{m \in [M]}$ denoting, e.g., their average monetary costs for processing one prompt. We assume $c^{(1)} \leq c^{(2)} \leq \dots \leq c^{(M)}$. Let $r: \mathcal{X} \rightarrow [M]$ be a *router* that, given a prompt, predicts the most suitable LLM. In the standard (“static”) routing problem, we seek a router which achieves (cf. Chen et al. [2023b], Dekoninck et al. [2025], Woisetschläger et al. [2025], Somerstep et al. [2025])

$$\min_{r: \mathcal{X} \rightarrow [M]} \sum_{m \in [M]} \mathbb{E}_{(\mathbf{x}, \mathbf{y})} \left[\mathbf{1}(r(\mathbf{x}) = m) \cdot \ell(\mathbf{x}, \mathbf{y}, h^{(m)}) \right] : \sum_{m \in [M]} \mathbb{E}_{(\mathbf{x}, \mathbf{y})} \left[\mathbf{1}(r(\mathbf{x}) = m) \cdot c^{(m)} \right] \leq B. \quad (1)$$

Here, $B \geq 0$ is some fixed budget on the cost of the routed solution. We use $h^{(m)}(\mathbf{x}) \doteq \text{predict}(p^{(m)}(\cdot | \mathbf{x}))$ for some fixed prediction function predict .

Model routing strategies. The most naïve routing strategy *randomly* assigns prompts to the various models, potentially pruning inadmissible solutions (see Appendix D). A more refined strategy is to *learn* a router. Hu et al. [2024b] proposed an intuitive strategy (see also Narasimhan et al. [2022]), wherein one constructs a predictor $\gamma^{(m)}: \mathcal{X} \rightarrow \mathbb{R}_+$ of the expected loss incurred by each LLM $h^{(m)}$, and routes via

$$r(\mathbf{x}) = \operatorname{argmin}_{m \in [M]} \left[\gamma^{(m)}(\mathbf{x}) + \lambda \cdot c^{(m)} \right]. \quad (2)$$

Here, $\lambda \geq 0$ is a hyper-parameter trading between cost and quality. In §5, we show formally that the routing rule in (2) is a plug-in estimator of a theoretically optimal routing rule (cf. also Somerstep et al. [2025]).

Given a training sample $S_{\text{tr}} \doteq \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N_{\text{tr}}}$, there are several approaches to construct γ . For example, given a text embedder $\varphi: \mathcal{X} \rightarrow \mathbb{R}^{D_p}$ (e.g., BERT [Devlin et al., 2019], Sentence-T5 [Ni et al., 2022]), one may employ:

$$\gamma_{\text{lin}}^{(m)}(\mathbf{x}) = \mathbf{w}_m^\top \varphi(\mathbf{x}) + b_m, \quad (3)$$

where $\mathbf{w}_m \in \mathbb{R}^{D_p}$, $b_m \in \mathbb{R}$ and (optionally) φ may be fit with a suitable empirical loss, e.g.,

$$\frac{1}{N} \sum_{i \in [N_{\text{tr}}]} \sum_{m \in [M]} (\ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h^{(m)}) - \gamma_{\text{lin}}^{(m)}(\mathbf{x}^{(i)}))^2. \quad (4)$$

A related matrix factorisation approach operating over *frozen* embeddings was proposed in Ong et al. [2025], fitted on a sample comprising of either *pairwise* comparisons [Ong et al., 2025] or pointwise correctness labels [Zhao et al., 2024a]. Alternatively, a K -NN estimator can be used [Hu et al., 2024b, Section 5.1]:

$$\gamma_{\text{kNN}}^{(m)}(\mathbf{x}) = \frac{1}{k} \sum_{i \in \text{NN}(\mathbf{x}, k)} \ell(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, h^{(m)}), \quad (5)$$

where $\text{NN}(\mathbf{x}, k)$ denotes the k -nearest neighbours of (the embeddings of) \mathbf{x} in S_{tr} .

3 Model Routing with a Dynamic LLM Pool

We now formalise the model routing problem when the set of LLMs may vary *dynamically*.

3.1 Problem Setup

The routing setup in (1) assumed a *static* pool of LLMs: indeed, observe that the linear model (3) only estimates parameters $\{(\mathbf{w}_m, b_m)\}_{m \in [M]}$ for a fixed set of LLMs, namely, $\{p^{(m)}\}_{m \in [M]}$. In practice, the LLM pool may frequently change, as new models are released and old models are deprecated. Naïvely, one may simply re-train

a router such as (3) with each such new LLM. However, this requires both annotating each training sample with the new LLM’s predictions, several steps of iterative training, and initiating a fresh router deployment. For a constantly refreshing LLM pool, this can impose a non-trivial overhead (e.g., computational cost). This motivates an alternate routing setup.

Concretely, let \mathcal{H}_{all} denote the set of all possible LLM predictors, where for simplicity 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}_{all} . Let $\mathcal{H}_{\text{tr}} = \{h_{\text{tr}}^{(1)}, \dots, h_{\text{tr}}^{(M)}\} \in \mathbb{H}$ denote the set of LLM predictors observed during training. During evaluation, we seek to route amongst the LLM predictors in $\mathcal{H}_{\text{te}} = \{h_{\text{te}}^{(1)}, \dots, h_{\text{te}}^{(N)}\} \in \mathbb{H}$. If $\mathcal{H}_{\text{tr}} = \mathcal{H}_{\text{te}}$, we obtain the original routing problem in (1). However, we aim to allow $\mathcal{H}_{\text{tr}} \neq \mathcal{H}_{\text{te}}$, including the case $\mathcal{H}_{\text{tr}} \cap \mathcal{H}_{\text{te}} = \emptyset$.

To accommodate such a dynamic LLM pool, we first modify our router to accept both an input prompt *and* a set of candidate LLMs, with the goal to pick the best option from this set; i.e., we consider *dynamic routers* $\mathcal{R} \doteq \{r(\cdot, \mathcal{H}): \mathcal{X} \rightarrow [|\mathcal{H}|] \mid \mathcal{H} \in \mathbb{H}\}$. Next, we assume that the set of training LLMs \mathcal{H}_{tr} is itself drawn from some *meta-distribution* \mathfrak{H} over \mathbb{H} . Rather than perform well on the *specific* set \mathcal{H}_{tr} , we would like to generalise to *any* set of LLMs drawn from \mathfrak{H} . Concretely, we wish to solve:

$$\min_{r \in \mathcal{R}} \mathbb{E} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \ell(\mathbf{x}, \mathbf{y}, h^{(m)}) \right] : \mathbb{E} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot c(h^{(m)}) \right] \leq B, \quad (6)$$

where as before $B \geq 0$ denotes a cost budget, $\mathcal{H} \doteq \{h^{(1)}, \dots, h^{(M)}\} \sim \mathfrak{H}$ denotes a sample of M LLMs, $c: \mathcal{H}_{\text{all}} \rightarrow \mathbb{R}_+$ denotes the cost of a given LLM, and \mathbb{E} is a shorthand for $\mathbb{E}_{(\mathbf{x}, \mathbf{y}, \mathcal{H})}$.

3.2 Optimal Routing with a Dynamic Pool

To guide the design of a dynamic router, we begin by studying the *Bayes-optimal* rule for (6). Interestingly, we find the Bayes-optimal rule *decomposes* across each of the constituent LLMs. The result is known for a *fixed* candidate set $\mathcal{H} \in \mathbb{H}$ [Jitkrittum et al., 2023, Lemma F.1], [Dekoninck et al., 2025, Somerstep et al., 2025]. The distinction arises for a *varying* candidate set, where the result closely mirrors Tailor et al. [2024, Eq. 6], derived in the related setting of learning to defer to an expert (see §6).

Proposition 1 (Optimal dynamic routing). *Under a mild regularity condition on \mathbb{P} , for any input $\mathbf{x} \in \mathcal{X}$, LLM candidate set $\mathcal{H} \in \mathbb{H}$, and budget $B > 0$, there exists a Lagrange multiplier $\lambda_{\mathfrak{H}} \geq 0$ such that the optimal dynamic router r^* for the constrained optimization in (6) is*

$$r^*(\mathbf{x}, \mathcal{H}) = \operatorname{argmin}_{m \in [|\mathcal{H}|]} \left[\mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})] + \lambda_{\mathfrak{H}} \cdot c(h^{(m)}) \right]. \quad (7)$$

Intuitively, it is optimal to route to the model that has the lowest expected loss on the given input \mathbf{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. If one wishes to *sweep* B to trace a cost-quality *deferral curve* (Appendix E.2), one may equally treat $\lambda_{\mathfrak{H}}$ as a constant to be swept from $[0, +\infty)$.

Special case: 0-1 Loss Consider a setting where an LLM response may be compared to a ground-truth target based on an exact match criteria: i.e., we pick the 0-1 loss $\ell(\mathbf{x}, \mathbf{y}, h^{(m)}) = \mathbf{1}[h^{(m)}(\mathbf{x}) \neq \mathbf{y}]$, with values either 0 (incorrect) or 1 (correct). Here, the optimal rule in (7) becomes:

$$\begin{aligned} r^*(\mathbf{x}, \mathcal{H}) &= \operatorname{argmin}_{m \in [|\mathcal{H}|]} \left[\gamma^*(\mathbf{x}, h^{(m)}) + \lambda_{\mathfrak{H}} \cdot c(h^{(m)}) \right] \\ \gamma^*(\mathbf{x}, h) &\doteq \mathbb{P}[\mathbf{y} \neq h(\mathbf{x}) \mid \mathbf{x}]. \end{aligned} \quad (8)$$

For simplicity, we will focus on the the 0-1 loss henceforth, and consider (8) as the optimal routing rule; our results can be readily adapted (as we shall see in our experiments in §7) to other loss functions by considering (7). Example problems where the 0-1 loss are appropriate include GSM8K [Cobbe et al., 2021], MMLU Hendrycks et al. [2021], and problems in SuperGLUE [Wang et al., 2019].

3.3 Plug-in Routing with a Dynamic Pool

Proposition 1 and (8) suggest a simple practical approach to routing with a dynamic pool of test LLMs $\mathcal{H}_{\text{te}} = \{h_{\text{te}}^{(n)}\}_{n \in [N]}$: we may construct a plug-in estimator $\gamma(\mathbf{x}, h)$ of $\gamma^*(\mathbf{x}, h)$, and estimate (8) via

$$r(\mathbf{x}, \mathcal{H}_{\text{te}}) = \operatorname{argmin}_{n \in [N]} \left[\gamma(\mathbf{x}, h_{\text{te}}^{(n)}) + \lambda \cdot c(h_{\text{te}}^{(n)}) \right]. \quad (9)$$

UniRoute: Routing with a Dynamic LLM Pool

- (1) **Train base router.** Fit parameters of γ_{uni} from (10) on training set $S_{\text{tr}} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N_{\text{tr}}}$.
- (2) **Embed test LLMs.** Compute $\Psi(h_{\text{te}})$ for each test LLM $h_{\text{te}} \in \mathcal{H}_{\text{te}}$, e.g., via (11) on a validation set $S_{\text{val}} = \{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})\}_{j=1}^{N_{\text{val}}}$.
- (3) **Route on new prompts.** Given a new input prompt x , pick the best test LLM via (9).

A key question arises: how should we parameterise $\gamma(x, h)$? We study this question next.

4 UniRoute: Universal Routing via an LLM Feature Representation

We present a general dynamic routing approach based on constructing *LLM feature representations*.

4.1 The UniRoute Approach

To enable routing with a dynamic LLM pool, we propose to parameterise γ by representing both prompts *and* LLMs as feature vectors. Specifically, for fixed $K > 1$ let $\Phi: \mathcal{X} \rightarrow \mathbb{R}^K$ and $\Psi: \mathcal{H}_{\text{all}} \rightarrow \mathbb{R}^K$ denote *feature maps* for prompts and LLMs respectively. Then, we propose:

$$\gamma_{\text{uni}}(\mathbf{x}, h) = \Phi(\mathbf{x})^\top \Psi(h). \quad (10)$$

We may now fit any parameters associated with Φ, Ψ on the training set S_{tr} , and then route as before via (9). Crucially, provided we define an easily computable Ψ , this seamlessly handles any $h \in \mathcal{H}_{\text{all}}$, *including one unobserved during training*; this is analogous to semantic output codes for zero-shot classification [Palatucci et al., 2009]. Thus, (10) provides an approach for *universal routing* with dynamic LLM pools.

To realise the potential of this approach, it now remains to specify $\Phi(\mathbf{x})$ and $\Psi(h)$.

Prompt representation. The choice of prompt representations $\Phi(\mathbf{x}) \in \mathbb{R}^K$ has been well-studied in prior work [Hu et al., 2024b]. A natural choice is to build on a general-purpose text embedding such as text-embedding-3 [OpenAI, 2025], NV-Embed [Lee et al., 2025], E5-Mistral-7B [Wang et al., 2024b], or Gecko [Lee et al., 2024b]. Such embeddings may be projected from a native D_{P} to $K \ll D_{\text{P}}$ dimensional space, e.g., via a linear transformation.

LLM representation. A good choice for $\Psi(h)$ is less apparent than that for $\Phi(\mathbf{x})$. Observe that the standard linear router (3) for a static pool $\mathcal{H}_{\text{tr}} = \{h_{\text{tr}}^{(1)}, \dots, h_{\text{tr}}^{(M)}\}$ corresponds to a *one-hot* LLM representation $\Psi_{\text{oh}}(h) = [\mathbf{1}(h = h_{\text{tr}}^{(m)})]_{m \in [M]}$, and a prompt representation $\Phi(\mathbf{x}) = \mathbf{W}\varphi(\mathbf{x}) \in \mathbb{R}^M$ for $\mathbf{W} \in \mathbb{R}^{M \times D_{\text{P}}}$. As noted previously, such an approach is inherently tied to the LLM pool \mathcal{H}_{tr} , analogous to the cold-start problem in collaborative filtering [Schein et al., 2002]. A naïve alternate idea is to flatten the LLM’s trained parameters. However, these would be in excess of billions of dimensions (exacerbating the risk of overfitting), and are inadmissible for many proprietary LLMs. We now examine an alternative LLM representation based on its performance on a subset of prompts.

4.2 Representing an LLM via the Prediction Error Vector

A key desideratum for our LLM representation Ψ is that $\Psi(h)^\top \Psi(h')$ ought to be large for a pair (h, h') of “similar” LLMs, and small for a pair of “dissimilar” LLMs. A reasonable definition of “similar” would thus enable the design of Ψ . To this end, we posit that two LLMs are “similar” if they have comparable performance on a set of representative prompts, akin to proposals in Thrush et al. [2024], Zhuang et al. [2024].

Concretely, suppose that we have access to a small (labelled) validation set $S_{\text{val}} = \{(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})\}_{j=1}^{N_{\text{val}}}$. Further, suppose that *any* LLM $h \in \mathcal{H}_{\text{all}} - \text{including new LLMs unobserved during training}$ – can be evaluated efficiently on these prompts. Then, one may represent the LLM based on its *prediction error vector* on prompts from S_{val} : for suitable $F: \mathbb{R}^{N_{\text{val}}} \rightarrow \mathbb{R}^K$ (e.g., a linear projection), we choose:

$$\Psi(h) = F \left([\mathbf{1}(\mathbf{y}^{(j)} \neq h(\mathbf{x}^{(j)}))]_{j \in [N_{\text{val}}]} \right) \in \mathbb{R}^K. \quad (11)$$

Interestingly, a special case of this is the K -NN method (5) from [Hu et al. \[2024b\]](#) applied to S_{val} : for $\Psi_{\text{knn}}(h) = [\mathbf{1}(\mathbf{y}^{(j)} \neq h(\mathbf{x}^{(j)}))]_{j \in [N_{\text{val}}]} \in \mathbb{R}^{N_{\text{val}}}$ and $\Phi_{\text{knn}}(\mathbf{x}) \in \{0, 1\}^{N_{\text{val}}}$ indicating which validation samples are the k -nearest neighbours of \mathbf{x} , (10) exactly reduces to (5). In general, however, it can prove useful to parameterise F and learn some compressed LLM representation in $K \ll N_{\text{val}}$ dimensions.

We remark that [Zhuang et al. \[2024\]](#) also considered representing LLMs as feature vectors, with the goal of enabling routing. Crucially, however, their representations do *not* enable generalisation to unseen LLMs, and thus do not support dynamic routing; cf. §6 for more discussion.

4.3 Discussion: UniRoute versus Standard Routing

A central assumption in UniRoute is that for any new LLM, one may efficiently compute $\Psi(\cdot)$; for $\Psi(\cdot)$ given by (11), this in turn assumes that any new LLM can be efficiently evaluated on S_{val} . We stress some important points regarding this. First, the choice of prompts in S_{val} is of clear import. In the simplest case, this may be a small subset of the training set S_{tr} . More generally, these could be hand curated based on domain knowledge, or drawn from a standard benchmark suite (which is often available for any new LLM). Second, we emphasise that S_{val} is assumed to be of modest size (e.g., $\mathcal{O}(10^3)$); consequently, performing inference for a new LLM on S_{val} is not prohibitive. Third, UniRoute involves *significantly less overhead* than naïve router re-training. Per §3.1, re-training a router such as (3) on S_{val} involves several steps of iterative training, which for a constantly refreshing LLM pool can impose a cumulatively onerous overhead. Further, since S_{val} is of modest size, re-training the router is susceptible to overfitting; thus, UniRoute can also yield better quality.

Interestingly, the K -NN method from [\[Hu et al., 2024b\]](#) – which, as noted in §4.2, is a special case of UniRoute – does support new LLMs without re-training. Indeed, one may readily compute γ_{knn} in (5) based solely on the prediction error vector. However, as S_{val} is assumed to be of modest size, this approach may not generalise favourably; indeed, even in moderate data regimes, [Zhuang et al. \[2024\]](#) observed that K -NN may underperform. Further, it does not exploit any information from the (potentially large) *training* set S_{tr} .

These limitations notwithstanding, K -NN has the appealing ability to exploit non-linear structure in the data. We now explore a *cluster-based* instantiation of UniRoute with a similar property.

5 UniRoute with Cluster-Based LLM Feature Representations

Building on the above, we now consider certain *cluster-based* LLM representations, involving either unsupervised or supervised cluster assignments based on a large set of training prompts.

5.1 Representing an LLM via Per-Cluster Prediction Errors

We propose an instantiation of (11) that represents any LLM $h \in \mathcal{H}_{\text{all}}$ through its average errors $\Psi_{\text{clust}}(h) \in [0, 1]^K$ on $K > 1$ pre-defined *clusters*. Similarly, we represent prompts via their cluster membership $\Phi_{\text{clust}}(\mathbf{x}) \in \{0, 1\}^K$. This yields the following instantiation of (10):

$$\gamma_{\text{clust}}(\mathbf{x}, h) \doteq \Phi_{\text{clust}}(\mathbf{x})^\top \Psi_{\text{clust}}(h). \quad (12)$$

Intuitively, $\gamma_{\text{clust}}(\mathbf{x}, h)$ estimates the performance of a given LLM on a prompt \mathbf{x} by examining its performance on *similar* prompts, i.e., those belonging to the same cluster.

Naïvely, one may directly cluster S_{val} ; however, this is prone to overfitting, since (by assumption) the set is of modest size. Thus, we instead cluster the prompts in the *training* set S_{tr} . We then use this to group the *validation* prompts into K disjoint clusters, and compute per-cluster errors for a new LM using S_{val} . Concretely, given a text embedder $\varphi: \mathcal{X} \rightarrow \mathbb{R}^{D_p}$, we compute $\Phi_{\text{clust}}(\mathbf{x}), \Psi_{\text{clust}}(h)$ via:

- (i) Cluster the *training* set embeddings $\{\varphi(\mathbf{x}^{(i)})\}_{i=1}^{N_{\text{tr}}}$ to construct K non-overlapping clusters. This yields a cluster assignment map $\Phi_{\text{clust}}: \mathcal{X} \rightarrow \{0, 1\}^K$, where the k -th index is 1 when \mathbf{x} belongs to cluster k (i.e., it is on average closest to samples in cluster k). This step does not require labels.
- (ii) Assign each *validation* set prompt to a cluster. Let $C_k \doteq \{(\mathbf{x}, \mathbf{y}) : (\mathbf{x}, \mathbf{y}) \in S_{\text{val}}, \Phi_{\text{clust},k}(\mathbf{x}) = 1\}$ be the subset of the validation set that belongs to cluster k .
- (iii) For any LLM $h \in \mathcal{H}_{\text{all}}$, compute $\Psi_{\text{clust}}(h) \in [0, 1]^K$ using its per-cluster validation errors:

$$\Psi_{\text{clust},k}(h) \doteq \frac{1}{|C_k|} \sum_{(\mathbf{x}, \mathbf{y}) \in C_k} \mathbf{1}[\mathbf{y} \neq h(\mathbf{x})]. \quad (13)$$

Plugging these into (12), we may now approximate the expected loss for $h_{\text{te}}^{(n)}$ on an input prompt \mathbf{x} using the average error of the LLM on the cluster the prompt is assigned to, and route via (9).

This cluster-based instantiation of UniRoute can route with new LLMs in a highly efficient manner: given any new test LLM, we simply need to estimate its average per-cluster clusters for all validation prompts. This does *not* require any expensive gradient updates, and is a *one-off cost*: further routing can operate entirely on this vector, and is oblivious to any further changes in the LLM pool.

A natural choice of clustering algorithm in step (ii) is K -means [MacQueen, 1967], which returns a set of K centroids and an assignment map Φ_{clust} that assigns prompts to the cluster with the nearest centroid. For $K = 1$, the router devolves to the *ZeroRouter* from [Hu et al., 2024b] (see Appendix D). Clearly, the practical utility of UniRoute depends on selection of K ; empirically, UniRoute is reasonably robust to this parameter. An illustration of our proposal is shown in Figure 1.

5.2 From Fixed to Learned Cluster Assignment Maps

The above cluster-based router does not leverage the labels in S_{tr} . We may exploit this information to further improve routing quality. Specifically, given the same clustering as above, we propose to *learn* a cluster assignment map $\Phi_{\text{clust}}(\cdot; \boldsymbol{\theta}) \in [0, 1]^K$ parameterised by $\boldsymbol{\theta}$, that can better map an input prompt to a distribution over clusters. Specifically, we parameterise $\Phi_{\text{clust},k}(\mathbf{x}; \boldsymbol{\theta}) \propto \exp(\boldsymbol{\theta}_k^\top \varphi(\mathbf{x}))$, for $\boldsymbol{\theta} \in \mathbb{R}^{K \times D_p}$ and text embedding φ . Analogous to (12), we have:

$$\gamma_{\text{clust}}(\mathbf{x}, h; \boldsymbol{\theta}) = \Phi_{\text{clust}}(\mathbf{x}; \boldsymbol{\theta})^\top \Psi_{\text{clust}}(h),$$

where $\Psi_{\text{clust}}(h)$ denotes the per-cluster errors for the LM estimated from the validation set S_{val} , as in (13); note that this does *not* depend on $\boldsymbol{\theta}$. To pick $\boldsymbol{\theta}$, we minimize the log loss on the training set S_{tr} against the correctness labels for the training LMs \mathcal{H}_{tr} :

$$-\sum_{(\mathbf{x}, \mathbf{y}) \in S_{\text{tr}}} \sum_{h \in \mathcal{H}_{\text{tr}}} \mathbf{1}[y \neq h(\mathbf{x})] \cdot \log \gamma_{\text{clust}}(\mathbf{x}, h; \boldsymbol{\theta}) + \mathbf{1}[y = h(\mathbf{x})] \cdot \log (1 - \gamma_{\text{clust}}(\mathbf{x}, h; \boldsymbol{\theta})).$$

5.3 Excess Risk Bound

We now present an excess risk bound for our cluster-based routing strategy. Suppose we represent the underlying data distribution over (\mathbf{x}, \mathbf{y}) by a mixture of K latent components: $\mathbb{P}(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^K \pi_k \cdot \mathbb{P}(\mathbf{x}, \mathbf{y} | z = k)$, where z denotes a latent random variable that identifies the mixture component and $\pi_k = \mathbb{P}(z = k)$. For a fixed k , we may denote the probability of incorrect predictions for $h \in \mathcal{H}$ conditioned on $z = k$ by: $\Psi_k^*(h) \doteq \mathbb{P}_{\mathbf{x}, \mathbf{y} | z=k} [h(\mathbf{x}) \neq \mathbf{y}]$. Then, cluster-based routing seeks to mimic the following population rule:

$$\begin{aligned} \tilde{r}^*(\mathbf{x}, \mathcal{H}_{\text{te}}) &= \operatorname{argmin}_{n \in [N]} \left[\gamma_{\text{clust}}^*(\mathbf{x}, h_{\text{te}}^{(n)}) + \lambda \cdot c(h_{\text{te}}^{(n)}) \right]; \\ \gamma_{\text{clust}}^*(\mathbf{x}, h_{\text{te}}^{(n)}) &= \sum_{k \in [K]} \mathbb{P}(z = k | \mathbf{x}) \cdot \Psi_k^*(h_{\text{te}}^{(n)}). \end{aligned} \quad (14)$$

Proposition 2. Let r^* be as per (8). For any $\mathcal{H}_{\text{te}} = \{h_{\text{te}}^{(n)}\}_{n \in [N]} \in \mathbb{H}$, $h_{\text{te}}^{(n)} \in \mathcal{H}_{\text{te}}$, and $\mathbf{x} \in \mathcal{X}$, let:

$$\Delta_k(\mathbf{x}, h_{\text{te}}^{(n)}) \doteq \left| \mathbb{P}_{\mathbf{y} | \mathbf{x}, z=k} [h_{\text{te}}^{(n)}(\mathbf{x}) \neq \mathbf{y}] - \Psi_k^*(h_{\text{te}}^{(n)}) \right|.$$

Let $R_{01}(r, \mathcal{H}_{\text{te}}) \doteq \sum_n \mathbb{P} [h_{\text{te}}^{(n)}(\mathbf{x}) \neq \mathbf{y} \wedge r(\mathbf{x}, \mathcal{H}_{\text{te}}) = m]$ denote the 0-1 risk. Then under a regularity condition on \mathbb{P} , the difference in 0-1 risk between \tilde{r}^* and r^* is bounded by:

$$\mathbb{E}_{\mathcal{H}_{\text{te}}} [R_{01}(\tilde{r}^*, \mathcal{H}_{\text{te}})] - \mathbb{E}_{\mathcal{H}_{\text{te}}} [R_{01}(r^*, \mathcal{H}_{\text{te}})] \leq \mathbb{E}_{\mathcal{H}_{\text{te}}, \mathbf{x}} \left[\max_{h_{\text{te}}^{(n)} \in \mathcal{H}_{\text{te}}, k \in [K]} \Delta_k(\mathbf{x}, h_{\text{te}}^{(n)}) \right].$$

This suggests that the quality gap between cluster-based routing in (14) and the optimal rule in (8) is bounded by the discrepancy between the per-cluster and per-prompt errors: i.e., the gap between the LLM's error on a prompt versus the average constituent cluster error (see Appendix C.3 for proof).

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. [2021]
RouteLLM	2	Pairwise comparison metrics.	✓	✓	✗	Ong et al. [2025]
K -NN router	Any	Pointwise evaluation, query embeddings.	✓	✓	△	Hu et al. [2024b], Shnitzer et al. [2023]
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 model routing approaches. Adaptive computation refers to the ability to trade quality for a reduced inference cost. \triangle : The K -NN approach considered in Hu et al. [2024b], Shnitzer et al. [2023] is for a fixed pool of LLMs. However, the approach can be straightforwardly extended to support unseen models at test time (i.e., dynamic pool).

6 Related Work

Model routing. Model routing has emerged as a simple yet effective strategy to lower LLMs’ inference cost [Hendy et al., 2023, Hari and 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, Aggarwal et al., 2024], k -nearest neighbours [Hu et al., 2024b, Shnitzer et al., 2023, Stripelis et al., 2024, Lee et al., 2024a], matrix factorisation [Ong et al., 2025, Zhuang et al., 2024, Li, 2025], 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., 2024b], and enhancing router robustness [Dann et al., 2024, Montreuil et al., 2025, Shafran et al., 2025]. 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 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 and Jones, 2001, Wang et al., 2018, Swayamdipta et al., 2018, Rawat et al., 2021, Wang et al., 2022, Kag et al., 2023, Jitkrittum et al., 2023]. Recently, cascades have also been successfully proven in the case of LLMs [Varshney and Baral, 2022, Chen et al., 2023b, Gupta et al., 2024, Yue et al., 2024, Chen et al., 2024a, Nie et al., 2024, Chuang et al., 2025].

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 and Wegkamp, 2008, Cortes et al., 2016], selective classification [Geifman and El-Yaniv, 2019, Narasimhan et al., 2024a,b], and learning to defer to an expert [Madras et al., 2018, Sangalli et al., 2023]. Following pioneering studies of Trapeznikov and Saligrama [2013], Bolukbasi et al. [2017], Mozannar and Sontag [2020], a series of works have studied the routing and cascading problem through these lenses [Narasimhan et al., 2022, Mao et al., 2024a,b,c,d].

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, Hu et al., 2024a], or producing a single fused model [Singh and 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 and 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].

Method	Dataset	EmbedLLM			RouterBench			Math+Code			SPROUT o3-mini		
		Area (50%) \uparrow	Area \uparrow	QNC \downarrow	Area (50%) \uparrow	Area \uparrow	QNC \downarrow	Area (50%) \uparrow	Area \uparrow	QNC \downarrow	Area (50%) \uparrow	Area \uparrow	QNC \downarrow
ZeroRouter [Hu et al., 2024b]		.285 *	.607 *	87.5% *	.320 *	.689 *	99.9%	.193 *	.395 *	82.8% *	.404 *	.820 *	100.0% *
K-NN Hu et al. [2024b], Shnitzer et al. [2023]		.298 *	.636 *	46.1% *	.328 *	.707 *	99.7%	.237 *	.487 *	25.7%	.418 *	.844 *	29.6% *
UniRoute (<i>K</i> -Means)		.307 *	.648 *	33.9%	.332	.712	99.4%	.238	.490	25.7%	.421	.850	19.6%
UniRoute (LearnedMap)		.308	.651	33.2%	.331	.711	99.6%	—	—	—	.420	.846	23.4%
MLP (Clairvoyant)		.314	.664	26.9%	.339	.723	95.2%	.242	.500	25.1%	.427	.859	4.5%

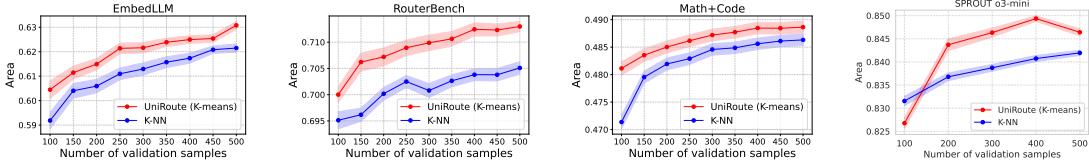


Figure 2: **Top:** We report the *area* under the deferral curve (up to 50% and 100% cost), and the Quality-Neutral Cost (QNC), i.e., the minimum relative cost to achieve the same performance as the most accurate LLM. For Math+Code, we do not have training LLMs; so we do not report results for UniRoute (LearnedMap). * indicates the method is statistically significantly worse than UniRoute (LearnedMap) at significance level $\alpha = 0.01$ (we compare against K-means for Math+Code). **MLP (Clairvoyant)** is an *oracle* that uses the test LLMs for training (provides a performance *upper bound*). **Bottom:** Areas under the deferral curve (\uparrow) with 96% CI on **unseen test LLMs** for varying number of validation samples. UniRoute (*K*-means) consistently outperforms *K*-NN for small sample sizes.

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, Zhou 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., 2024a, Cai et al., 2024b, Li et al., 2024d,e]. Recent works have studied approaches to combine speculative decoding with early-exits [Elhoushi et al., 2024] and cascades [Narasimhan et al., 2025].

7 Experiments

We demonstrate the effectiveness of UniRoute in the setting of observing **new LLMs at test time** on EmbedLLM [Zhuang et al., 2024], RouterBench [Hu et al., 2024b], a Math+Code dataset from Dekoninck et al. [2025] (containing a subset of Minerva Math and LiveCodeBench [Lewkowycz et al., 2022, Jain et al., 2024]), SPROUT o3-mini [Somerstep et al., 2025], and Chatbot Arena [Zheng et al., 2023].

7.1 Experimental Setup

We first describe the **LLM pool construction**. With EmbedLLM, RouterBench, and SPROUT o3-mini, we partition the set of LLMs into two disjoint sets: training models (\mathcal{H}_{tr} in §5) and testing models (\mathcal{H}_{te}). For EmbedLLM (112 LLMs) and SPROUT o3-mini (15 LLMs), we use a random subset of $2/3$ for training and $1/3$ for testing. For RouterBench (11 LLMs in total), we use a random 50% for training and the rest for testing. For Math+Code, we use all 4 LLMs for testing; consequently, the training data is unlabeled.

For each of the 400 independent trials, we randomly split examples into 60%/10%/30% for training, validation, and testing (for RouterBench, and SPROUT o3-mini we use 1% for validation). The training portion is for training a router, and only has correctness labels of training models. The validation split is the small dataset used to represent each test-time LLM as a feature vector (see §5.1). All baselines are evaluated on the test examples and *only* on the test LLMs. We use Gecko 1B [Lee et al., 2024b] to produce a 768-dimensional prompt embedding where required.

Per-example metrics. All datasets considered in the main text use binary accuracy as the the evaluation metric. Thus, all methods rely on the deferral rule described in (8).

Baselines. We reiterate that, with the notable exception of *K*-NN [Hu et al., 2024b], **most existing routers in the literature are inadmissible** in a settingwith a *dynamic* pool of LLMs. This is true of the multi-layer perceptron (MLP) [Hu et al., 2024b], matrix factorization [Ong et al., 2025, Zhuang et al., 2024] and BERT

[Ong et al., 2025, Ding et al., 2024] routers, which have a *fixed* number of outputs, one per training LLM, and are thus inherently tied to those LLMs. Nonetheless, we include some of these methods as an *oracle* (by assuming the set of LLMs is fully known) to estimate an *upper bound* on router performance on unseen LLMs. The baselines we consider are:

- *ZeroRouter* Hu et al. [2024b]. A *random router* that randomizes between two LLMs, where the LLMs and mixing coefficients are chosen to maximize the expected quality on the validation sample, while satisfying the budget constraint (details in Appendix D).
- *K-NN* [Hu et al., 2024b, Shnitzer et al., 2023]. A special case of UniRoute (see §4.2) that for each prompt, looks up the K nearest prompts in the validation set in the space of prompt embeddings, computes γ for each test LLM using (5) (with the 0-1 loss), and routes via (2).
- *MLP (Clairvoyant upper bound)*. An MLP router with one output for each train and test LLM, trained on prompt embeddings to estimate γ . For training, we use the combined training and validation set, annotated with correctness labels from both the train and test LLMs. This oracle baseline provides an estimate of the *performance achievable when all LLMs are observed*.

We compare them to our UniRoute cluster-based routing method using both (i) unsupervised K -means for clustering (§5.1), and (ii) the supervised learned cluster assignment map (§5.2). In Appendix F, we also include as a baseline *Clairvoyant version of the matrix factorization router* [Ong et al., 2025, Zhuang et al., 2024].

Evaluation. We evaluate each method with a *deferral curve* (Appendix E.2; [Jitkrittum et al., 2023, Wang et al., 2024a, Hu et al., 2024b]), which plots the trade-off of average quality against cost. The trade-off is realized by varying $\lambda_{\mathcal{S}}$ in the routing rule in (7). For EmbedLLM, we use the number of parameters of the LLM as the cost of processing one prompt. This cost is a proxy for the amount of computation required to call each LLM. For RouterBench and SPROUT o3-mini, we plot LLMs’ API calling costs (in USD) as available in the dataset.

Hyper-parameter tuning. We apply the following procedure to pick K for K -NN, K -means, and the Learned cluster map: for each parameter, 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. This evaluation metric can be seen as the average improvement in quality per unit cost (analogous to the AIQ metric used in Hu et al. [2024b]). The parameter with the maximum area is then chosen. See Appendix F.1 for details.

7.2 Experimental Results

We present deferral curves for different methods on EmbedLLM in Figure 3, and on other datasets in Appendix F.2. Each isolated point \times 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 area up to 50% cost; and (iii) the quality-neutral cost (QNC) or the minimum relative cost needed to achieve the same performance as the most accurate testing LLM. Note that the QNC is analogous to the call-performance threshold reported in Ong et al. [2025]. The table in Figure 2 summarizes these metrics for all four datasets.

UniRoute enables generalisation under dynamic LLM pools. UniRoute yields competitive quality-cost trade-offs, with the gains over K -NN being *statistically significant*. In particular, on EmbedLLM – featuring > 30 **unseen LLMs** – we provide compelling gains over K -NN on all metrics. Further, on all four datasets, we consistently outperform ZeroRouter, which was noted as a strong baseline in Hu et al. [2024b]. Appendix F.3 further shows UniRoute is effective when the LLM representations are constructed from Chatbot Arena, and evaluated on EmbedLLM.

UniRoute is effective even with a small validation sample. We show in Figure 2 (bottom), that our proposal is often significantly better than K -NN across a range of validation sample sizes. One potential reason for this is the requirement in K -NN 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 exploit the training data either in an unsupervised (K -means) or supervised (Learned cluster map) manner.

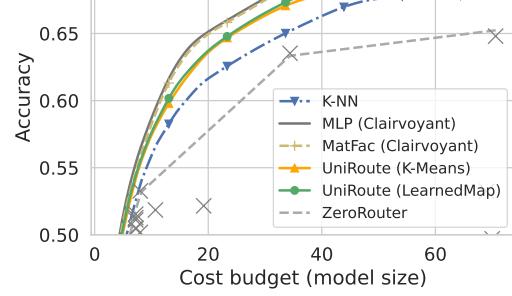


Figure 3: Deferral curves on EmbedLLM.

UniRoute is robust to choice of clusters K . Appendix F shows that in general, UniRoute is effective for several different K values, and our hyperparameter selector picks an effective K .

UniRoute maintains generalisation under static LLM pools. While the dynamic pool setting is the focal point of our work, we show in Appendix F.4 that even in the *static* LLM pool setting, UniRoute typically performs comparable to most baselines.

Qualitative analysis of $\Psi(h)$ embeddings. Appendix F.5 presents further qualitative analysis of the LLM embeddings deriving from the prediction error vector representation. These show that a largely intuitive grouping of “similar” LLMs (e.g., coding-specialists) in the embedding space.

8 Conclusion and Future Work

We present principled strategies for routing amongst multiple unseen test-time LLMs, by leveraging a *prediction error vector* LLM representation. An interesting future direction is to enhance routing robustness to prompt distribution shifts, such as by allowing the set of representative prompts S_{val} to dynamically vary. Such a routing system will further reduce the need for frequent router re-training.

References

- Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=3zKtaqxLhW>.
- Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappagantu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam . Automix: Automatically mixing language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL <https://openreview.net/forum?id=e6WrwIvgzX>.
- Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhai Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. PaLM 2 technical report, 2023.
- 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.

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

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.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS ’23*, Red Hook, NY, USA, 2023. Curran Associates Inc.

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.

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

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *Forty-second International Conference on Machine Learning*, 2025. URL <https://openreview.net/forum?id=U08mUogGDM>.

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

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

Christoph Dann, Yishay Mansour, Teodor Vanislavov Marinov, and Mehryar Mohri. Domain adaptation for robust model routing. In *Adaptive Foundation Models: Evolving AI for Personalized and Efficient Learning*, 2024. URL <https://openreview.net/forum?id=86eGohKPyy>.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiaoshi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shufeng Zhou, Size Zheng, T. Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L.

Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Liu, Xin Xie, Xingkai Yu, Xinnan Song, Xinyi Zhou, Xinyu Yang, Xuan Lu, Xuecheng Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Zheng, Yichao Zhang, Yiliang Xiong, Yilong Zhao, Ying He, Ying Tang, Yishi Piao, Yixin Dong, Yixuan Tan, Yiyuan Liu, Yongji Wang, Yongqiang Guo, Yuchen Zhu, Yuduan Wang, Yuheng Zou, Yukun Zha, Yunxian Ma, Yuting Yan, Yuxiang You, Yuxuan Liu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhewen Hao, Zhihong Shao, Zhiniu Wen, Zhipeng Xu, Zhongyu Zhang, Zhusu Li, Zihan Wang, Zihui Gu, Zilin Li, and Ziwei Xie. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024. URL <https://arxiv.org/abs/2405.04434>.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs, 2025. URL <https://arxiv.org/abs/2410.10347>.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.

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)*, page 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)*, pages 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*, pages 94–104, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL <https://aclanthology.org/W17-4912>.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10323–10337. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/frantar23a.html>.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2151–2159. PMLR, 09–15 Jun 2019.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhota, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenjin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena

Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madiam Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghatham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaoqian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

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>.

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

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>.

Jinwu Hu, Yufeng Wang, Shuhai Zhang, Kai Zhou, Guohao Chen, Yu Hu, Bin Xiao, and Mingkui Tan. Dynamic ensemble reasoning for llm experts, 2024a. URL <https://arxiv.org/abs/2412.07448>.

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*, 2024b. URL <https://openreview.net/forum?id=IVXmV8Uxwh>.

Zhongzhan Huang, Guoming Ling, Vincent S. Liang, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. Routereval: A comprehensive benchmark for routing llms to explore model-level scaling up in LLMs, 2025. URL <https://arxiv.org/abs/2503.10657>.

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.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

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, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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, pages 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>.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. OrchestraLLM: Efficient orchestration of language models for dialogue state tracking. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1434–1445, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.nacl-long.79. URL <https://aclanthology.org/2024.nacl-long.79>.

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, 2024b. 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*, pages 19274–19286. PMLR, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Llm inference serving: Survey of recent advances and opportunities, 2024a. URL <https://arxiv.org/abs/2407.12391>.

Tianle Li, Wei-Lin Chiang, and Lisa Dunlap. Introducing hard prompts category in Chatbot Arena. <https://lmsys.org/blog/2024-05-17-category-hard>, 2024b.

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, 2024c. 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, 2024d.

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, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA, November 2024e. 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, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 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, page 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. Principled approaches for learning to defer with multiple experts. In Reneta P. Barneva, Valentin E. Brimkov, Claudio Gentile, and Aldo Pacchiano, editors, *Artificial Intelligence and Image Analysis*, pages 107–135, Cham, 2024b. Springer Nature Switzerland. ISBN 978-3-031-63735-3.

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*, 2024c. URL <https://openreview.net/forum?id=0c02XakUJK>.

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, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 34738–34759. PMLR, 21–27 Jul 2024d. URL <https://proceedings.mlr.press/v235/mao24d.html>.

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Comput. Surv.*, 55(12), mar 2023. ISSN 0360-0300. doi: 10.1145/3578938. URL <https://doi.org/10.1145/3578938>.

Alireza Mohammadshahi, Arshad Rafiq Shaikh, and Majid Yazdani. Routoo: Learning to route to large language models effectively, 2024.

Yannis Montreuil, Axel Carlier, Lai Xing Ng, and Wei Tsang Ooi. Adversarial robustness in two-stage learning-to-defer: Algorithms and guarantees, 2025. URL <https://arxiv.org/abs/2502.01027>.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 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, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_jg6Sf6tuF7.

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*, 2024a. 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*, 2024b. URL <https://openreview.net/forum?id=DASh78rJ7g>.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=vo9t20wsmd>.

Jerzy Neyman and Egon Sharpe Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yafei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.146. URL <https://aclanthology.org/2022.findings-acl.146>.

Lunyu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. In *Proceedings of the 41st International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. Zero-shot learning with semantic output codes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, NIPS’09, page 1410–1418, Red Hook, NY, USA, 2009. Curran Associates Inc. ISBN 9781615679119.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *MLSys*, 2023. URL https://proceedings.mlsys.org/paper_files/paper/2023/hash/c4be71ab8d24cdfb45e3d06dbfca2780-Abstract-mlsys2023.html.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

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. *arXiv preprint arXiv:2110.10305*, 2021.

Ankit Singh Rawat, Veeranjaneyulu Sadhanala, Afshin Rostamizadeh, Ayan Chakrabarti, Wittawat Jitkrittum, Vladimir Feinberg, Seungyeon Kim, Hrayr Harutyunyan, Nikunj Saunshi, Zachary Nado, Rakesh Shivanna, Sashank J. Reddi, Aditya Krishna Menon, Rohan Anil, and Sanjiv Kumar. A little help goes a long way: Efficient LLM training by leveraging small LMs, 2024. URL <https://arxiv.org/abs/2410.18779>.

Aishwarya P S, Pranav Ajit Nair, Yashas Samaga, Toby Boyd, Sanjiv Kumar, Prateek Jain, and Praneeth Netrapalli. Tandem Transformers for inference efficient LLMs, 2024. URL <https://arxiv.org/abs/2402.08644>.

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.

Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’02, page 253–260, New York, NY, USA, 2002. Association for Computing Machinery. ISBN 1581135610. doi: 10.1145/564376.564421. URL <https://doi.org/10.1145/564376.564421>.

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, editors, *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=uLYc4L3C81A>.

Avital Shafran, Roei Schuster, Thomas Ristenpart, and Vitaly Shmatikov. Rerouting LLM routers, 2025. URL <https://arxiv.org/abs/2501.01818>.

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, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 22045–22055. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/fb2697869f56484404c8ceee2985b01d-Paper.pdf.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. Carrot: A cost aware rate optimal router, 2025. URL <https://arxiv.org/abs/2502.03261>.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from LLM routing. In Shabnam Tafreshi, Arjun Akula, João Sedoc, Aleksandr Drozd, Anna Rogers, and Anna Rumshisky, editors, *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 124–134, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

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 Diliphai 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*, pages 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>.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023.

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, editors, *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, pages 581–589, Scottsdale, Arizona, USA, 29 Apr–01 May 2013. PMLR.

Neeraj Varshney and Chitta Baral. Model cascading: Towards jointly improving efficiency and accuracy of NLP systems. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11007–11021, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fdb053c1c4a845aa-Paper.pdf.

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, pages 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*, page 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>.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, Mosharaf Chowdhury, and Mi Zhang. Efficient large language models: A survey. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL <https://openreview.net/forum?id=bsCCJHb08A>. Survey Certification.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: a stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

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=Mv02t0vbs4->.
- 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, editors, *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 580–590. AUAI Press, 2018.
- Yiding Wang, Kai Chen, Haisheng Tan, and Kun Guo. Tabi: An efficient multi-level inference system for large language models. In *Proceedings of the Eighteenth European Conference on Computer Systems*, EuroSys ’23, page 233–248, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394871. doi: 10.1145/3552326.3587438. URL <https://doi.org/10.1145/3552326.3587438>.
- Herbert Woisetschläger, Ryan Zhang, Shiqiang Wang, and Hans-Arno Jacobsen. Dynamically learned test-time model routing in language model zoos with service level guarantees, 2025. URL <https://arxiv.org/abs/2505.19947>.
- 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*, pages 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>.
- Zesen Zhao, Shuwei Jin, and Z Morley Mao. Eagle: Efficient training-free router for multi-llm inference. *arXiv preprint arXiv:2409.15518*, 2024a.
- 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, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4447–4462, Bangkok, Thailand, August 2024b. 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>.
- Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. BERT loses patience: Fast and robust inference with early exit. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/d4dd111a4fd973394238aca5c05bebe3-Paper.pdf.
- 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, editors, *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*, 2024a. URL <https://openreview.net/forum?id=rsY6J3ZaTF>.
- Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A survey on efficient inference for large language models, 2024b. URL <https://arxiv.org/abs/2404.14294>.

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>.

Appendix

Table of Contents

A Limitations	23
B Societal Impact	23
C Proofs of Results in Main Body	24
C.1 Intermediate Results	24
C.2 Proof of Proposition 1	26
C.3 Proof of Proposition 2	27
D Zero Routing	29
D.1 Special Case: Optimal Cluster Routing Rule for $K = 1$	29
E Experimental Setup	30
E.1 Splitting Data and LLMs	30
E.2 Evaluation: Deferral Curve	30
E.3 Implementation Details of UniRoute (LearnedMap)	30
E.4 Router Cost	32
F Additional Experimental Results	33
F.1 Hyper-parameter Choices & Statistical Significance	33
F.2 Deferral Curves and Additional Comparisons	34
F.3 Train on Chatbot Arena and Test on EmbedLLM	35
F.4 Static LLM Pool Setting	36
F.5 Visualisation of LLM Embeddings	37

A Limitations

Our work has some limitations that would be worthy subjects for future research. First, in the fully static LLM pool setting, our proposed method is not *guaranteed* to recover the performance of existing methods such as the linear router (3) (owing to a dependence on the validation sample, which may be a subset of the training data); designing *hybrid* static-dynamic routers would be of interest.

Second, while we proposed certain natural cluster-based routers, there could be other instances of UniRoute that are also worth exploring. A more systematic study of this design space would be worthwhile.

B Societal Impact

Our primary contribution is a mathematical formalism for routing amongst multiple dynamic LLMs, with concrete instantiations based on prompt clustering. These are coupled with empirical results on public benchmarks, typically involving LLM accuracy as the primary metric.

We do not foresee immediate negative societal impact from our mathematical framework. While well-studied in the literature, the underlying routing problem itself *could* lend itself to some undesirable outcomes; e.g., a router may overly favour models that produce outputs that are systematically biased on certain data sub-groups. We believe that our framework is amenable to such constraints, and believe that accounting for these is a worthy subject of future research.

C Proofs of Results in Main Body

In what follows, we use $r_{\mathcal{H}}(\mathbf{x})$ and $r(\mathbf{x}, \mathcal{H})$ interchangeably. As a shorthand, we define

$$S(r) \doteq \mathbb{E}_{(\mathbf{x}, \mathbf{y}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \ell(\mathbf{x}, \mathbf{y}, h^{(m)}) \right],$$

$$C(r) \doteq \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot c(h^{(m)}) \right].$$

The constrained optimization problem in (6) is equivalent to

$$\min_{r \in \mathcal{R}} S(r) \text{ subject to } C(r) \leq B. \quad (15)$$

It will be useful to consider the following related unconstrained optimization objective:

$$L(r, \lambda) = S(r) + \lambda \cdot C(r). \quad (16)$$

For any $\lambda \geq 0$, let r_λ^* be the minimiser of $L(r, \lambda)$.

C.1 Intermediate Results

We first provide results that will be useful for providing the main claims in §C.2 and §C.3.

Lemma 3. *Given $\lambda \geq 0$, the optimal solution $r_\lambda^* \in \arg \min_r L(r, \lambda)$ to the unconstrained problem in (16) is*

$$r_\lambda^*(\mathbf{x}, \mathcal{H}) \in \operatorname{argmin}_{m \in [|\mathcal{H}|]} \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})] + \lambda \cdot c(h^{(m)}).$$

Proof. Starting with the definition of L ,

$$\begin{aligned} L(r, \lambda) &= S(r) + \lambda \cdot C(r) \\ &= \mathbb{E}_{(\mathbf{x}, \mathbf{y}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \ell(\mathbf{x}, \mathbf{y}, h^{(m)}) \right] + \lambda \cdot \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot c(h^{(m)}) \right] \\ &\stackrel{(a)}{=} \mathbb{E}_{\mathcal{H}} \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{y}|\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \left\{ \ell(\mathbf{x}, \mathbf{y}, h^{(m)}) + \lambda \cdot c(h^{(m)}) \right\} \mid \mathbf{x} \right] \\ &= \mathbb{E}_{\mathcal{H}} \mathbb{E}_{\mathbf{x}} \underbrace{\left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \left\{ \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)}) \mid \mathbf{x}] + \lambda \cdot c(h^{(m)}) \right\} \right]}_{\mathcal{L}_{\mathcal{H}, \mathbf{x}}}, \end{aligned}$$

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

$$r^*(\mathbf{x}, \mathcal{H}) \in \operatorname{argmin}_{m \in [|\mathcal{H}|]} \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})] + \lambda \cdot c(h^{(m)}).$$

□

Lemma 4. *For any $\lambda \geq 0$, let r_λ^* be the minimiser of $L(r, \lambda)$. Then, for any r , if $C(r_\lambda^*) \geq C(r)$, then $S(r_\lambda^*) \leq S(r)$.*

Proof. Since r_λ^* minimises $L(r, \lambda)$, for any r , we have $L(r_\lambda^*, \lambda) \leq L(r, \lambda)$. That is,

$$\begin{aligned} S(r_\lambda^*) + \lambda \cdot C(r_\lambda^*) &\leq S(r) + \lambda \cdot C(r) \\ \iff S(r_\lambda^*) &\leq S(r) + \lambda \cdot (C(r) - C(r_\lambda^*)). \end{aligned} \quad (17)$$

Since $\lambda \geq 0$, it follows that $S(r_\lambda^*) \leq S(r)$ by the assumption that $C(r) - C(r_\lambda^*) \leq 0$. □

Lemma 5. Let r_λ^* be the minimiser of $L(r, \lambda)$. For $\lambda, \lambda' \geq 0$, $\lambda \geq \lambda'$ if and only if $C(r_\lambda^*) \leq C(r_{\lambda'}^*)$. In other words, $\lambda \mapsto C(r_\lambda^*)$ is a non-increasing function. Hence, $\sup_{\lambda \geq 0} C(r_\lambda^*) = C(r_0^*)$.

Proof. Since $r_\lambda^* \in \arg \min_r L(r, \lambda)$, by definition, we have

$$S(r_\lambda^*) + \lambda \cdot C(r_\lambda^*) \leq S(r) + \lambda \cdot C(r),$$

for any r , including $r_{\lambda'}^*$. This means

$$S(r_\lambda^*) + \lambda \cdot C(r_\lambda^*) \leq S(r_{\lambda'}^*) + \lambda \cdot C(r_{\lambda'}^*).$$

In a symmetric manner,

$$S(r_{\lambda'}^*) + \lambda' \cdot C(r_{\lambda'}^*) \leq S(r_\lambda^*) + \lambda' \cdot C(r_\lambda^*).$$

Adding the above two inequalities, we have

$$\begin{aligned} S(r_\lambda^*) + \lambda \cdot C(r_\lambda^*) + S(r_{\lambda'}^*) + \lambda' \cdot C(r_{\lambda'}^*) &\leq S(r_{\lambda'}^*) + \lambda \cdot C(r_{\lambda'}^*) + S(r_\lambda^*) + \lambda' \cdot C(r_\lambda^*) \\ &\implies \lambda \cdot C(r_\lambda^*) + \lambda' \cdot C(r_{\lambda'}^*) \leq \lambda \cdot C(r_{\lambda'}^*) + \lambda' \cdot C(r_\lambda^*) \\ &\implies \lambda \cdot (C(r_\lambda^*) - C(r_{\lambda'}^*)) + \lambda' \cdot (C(r_{\lambda'}^*) - C(r_\lambda^*)) \leq 0 \\ &\implies (\lambda - \lambda') (C(r_\lambda^*) - C(r_{\lambda'}^*)) \leq 0. \end{aligned}$$

The last inequality above implies that $\lambda \geq \lambda'$ if and only if $C(r_\lambda^*) \leq C(r_{\lambda'}^*)$. \square

Lemma 6. Let $\lambda, \lambda' \geq 0$ such that $\lambda \leq \lambda'$. Given $\mathbf{x} \in \mathcal{X}$ and $\mathcal{H} \in \mathbb{H}$, if $r_\lambda^*(\mathbf{x}, \mathcal{H}) \neq r_{\lambda'}^*(\mathbf{x}, \mathcal{H})$, then there exists a model pair $h^{(m)}, h^{(k)} \in \mathcal{H}$ with $h^{(m)} \neq h^{(k)}$ such that

$$\lambda \leq \frac{\mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(k)})] - \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})]}{c(h^{(m)}) - c(h^{(k)})} \leq \lambda'.$$

Proof. Given \mathcal{H} , define $A_m(\lambda, \mathbf{x}) \doteq \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})] + \lambda \cdot c(h^{(m)})$. Recall that

$$r_\lambda^*(\mathbf{x}, \mathcal{H}) = \operatorname{argmin}_{m \in [|\mathcal{H}|]} A_m(\lambda, \mathbf{x}).$$

Observe that given $(\mathbf{x}, \mathcal{H})$, for each $m \in [|\mathcal{H}|]$, $A_m(\lambda, \mathbf{x})$ is a one-dimensional affine function of λ . So, $r_\lambda^*(\mathbf{x}, \mathcal{H})$ is the index of the affine function that gives the lowest value when evaluated at λ . Fix \mathbf{x} . Since $\lambda \mapsto A_m(\lambda, \mathbf{x})$ is continuous, for any m , if $r_\lambda^*(\mathbf{x}, \mathcal{H}) \neq r_{\lambda'}^*(\mathbf{x}, \mathcal{H})$, it must mean that the index of the lowest affine function changes as we move from λ to λ' . This implies that there is an index pair m and k (with $m \neq k$) for which $\lambda \mapsto A_m(\lambda, \mathbf{x})$ and $\lambda \mapsto A_k(\lambda, \mathbf{x})$ cross, as we move from λ to λ' . The cross point is precisely at $\bar{\lambda}$ such that

$$\begin{aligned} A_m(\bar{\lambda}, \mathbf{x}) &= A_k(\bar{\lambda}, \mathbf{x}) \\ &\iff \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})] + \bar{\lambda} \cdot c(h^{(m)}) = \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(k)})] + \bar{\lambda} \cdot c(h^{(k)}) \\ &\iff \bar{\lambda} = \frac{\mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(k)})] - \mathbb{E}_{\mathbf{y}|\mathbf{x}} [\ell(\mathbf{x}, \mathbf{y}, h^{(m)})]}{c(h^{(m)}) - c(h^{(k)})}. \end{aligned}$$

\square

Lemma 7. Assume \mathbf{x} is a continuous random vector. Then, $\lambda \mapsto C(r_\lambda^*)$ is continuous.

Proof. We will show that for any $\Delta\lambda \in \mathbb{R}$, $\lim_{\Delta\lambda \rightarrow 0} |C(r_{\lambda+\Delta\lambda}^*) - C(r_\lambda^*)| = 0$. Observe that

$$\begin{aligned} |C(r_{\lambda+\Delta\lambda}^*) - C(r_\lambda^*)| &= \left| \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[c(h^{(r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}))}) \right] - \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[c(h^{(r_\lambda^*(\mathbf{x}, \mathcal{H}))}) \right] \right| \\ &\stackrel{(a)}{\leq} \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left| c(h^{(r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}))}) - c(h^{(r_\lambda^*(\mathbf{x}, \mathcal{H}))}) \right| \\ &= \mathbb{E}_{\mathcal{H}} \int_{\mathbf{x} \in \mathcal{X}} \left| c(h^{(r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}))}) - c(h^{(r_\lambda^*(\mathbf{x}, \mathcal{H}))}) \right| p(\mathbf{x}) d\mathbf{x} \doteq \spadesuit, \end{aligned}$$

where we applied Jensen's inequality at (a). Define $\mathcal{S}_{\Delta\lambda}$ to be the set of input points for which r_λ^* and $r_{\lambda+\Delta\lambda}^*$ make different decisions. Precisely,

$$\mathcal{S}_{\Delta\lambda}(\mathcal{H}) \doteq \{ \mathbf{x} \mid r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}) \neq r_\lambda^*(\mathbf{x}, \mathcal{H}) \}.$$

Let $c_{\max} \doteq \max_{h \in \mathcal{H}_{\text{all}}} c(h)$ i.e., the maximum per-query cost of any LLM in our finite universe. The quantity ♠ can be bounded further with

$$\begin{aligned} \spadesuit &= \mathbb{E}_{\mathcal{H}} \int_{x \in \mathcal{S}_{\Delta\lambda}(\mathcal{H})} \left| c\left(h^{(r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}))}\right) - c\left(h^{(r_{\lambda}^*(\mathbf{x}, \mathcal{H}))}\right) \right| p(\mathbf{x}) d\mathbf{x} \\ &\leq 2c_{\max} \mathbb{E}_{\mathcal{H}} \int_{x \in \mathcal{S}_{\Delta\lambda}(\mathcal{H})} p(\mathbf{x}) d\mathbf{x} \\ &= 2c_{\max} \mathbb{E}_{\mathcal{H}} \mathbb{P}(\mathcal{S}_{\Delta\lambda}(\mathcal{H})). \end{aligned}$$

The proof now amounts to showing that $\lim_{\Delta\lambda \rightarrow 0} \mathbb{E}_{\mathcal{H}} \mathbb{P}(\mathcal{S}_{\Delta\lambda}(\mathcal{H})) = 0$.

Define a shorthand $\lambda_{m,k}(\mathbf{x}, \mathcal{H}) \doteq \frac{\mathbb{E}_{\mathbf{y}|\mathbf{x}}[\ell(\mathbf{x}, \mathbf{y}, h^{(k)})] - \mathbb{E}_{\mathbf{y}|\mathbf{x}}[\ell(\mathbf{x}, \mathbf{y}, h^{(m)})]}{c(h^{(m)}) - c(h^{(k)})}$ where $h^{(m)}, h^{(k)} \in \mathcal{H}$. We start by expanding $\mathcal{S}_{\Delta\lambda}(\mathcal{H})$ to have

$$\begin{aligned} \mathcal{S}_{\Delta\lambda}(\mathcal{H}) &= \{\mathbf{x} \mid r_{\lambda+\Delta\lambda}^*(\mathbf{x}, \mathcal{H}) \neq r_{\lambda}^*(\mathbf{x}, \mathcal{H})\} \\ &\stackrel{(a)}{\subseteq} \{\mathbf{x} \mid \exists k \neq m : \lambda_{m,k}(\mathbf{x}, \mathcal{H}) \in [\lambda, \lambda + \Delta\lambda]\} \\ &= \bigcup_{m \neq k} \{\mathbf{x} \mid \lambda_{m,k}(\mathbf{x}, \mathcal{H}) \in [\lambda, \lambda + \Delta\lambda]\} \\ &\doteq \mathcal{F}_{\Delta\lambda}(\mathcal{H}), \end{aligned}$$

where at (a) the subset relationship is due to Lemma 6. Since $\mathcal{S}_{\Delta\lambda}(\mathcal{H}) \subseteq \mathcal{F}_{\Delta\lambda}(\mathcal{H})$, we have

$$\begin{aligned} \spadesuit &\leq 2c_{\max} \mathbb{E}_{\mathcal{H}} \mathbb{P}(\mathcal{F}_{\Delta\lambda}(\mathcal{H})) \\ &\stackrel{(a)}{\leq} 2c_{\max} \mathbb{E}_{\mathcal{H}} \sum_{m \neq k} \mathbb{P}(\{\mathbf{x} \mid \lambda_{m,k}(\mathbf{x}, \mathcal{H}) \in [\lambda, \lambda + \Delta\lambda]\}) \\ &= 2c_{\max} \mathbb{E}_{\mathcal{H}} \sum_{m \neq k} \mathbb{P}(\lambda_{m,k}(\mathbf{x}, \mathcal{H}) \in [\lambda, \lambda + \Delta\lambda]), \end{aligned}$$

where at (a) we use the union bound. Note that there are a finite number of summands because \mathcal{H}_{all} is finite. Since \mathbf{x} is a continuous random variable, for any \mathcal{H} , $\lambda_{m,k}(\mathbf{x}, \mathcal{H})$ is also a continuous random variable. As $\Delta\lambda \rightarrow 0$, we have $\mathbb{P}(\lambda_{m,k}(\mathbf{x}, \mathcal{H}) \in [\lambda, \lambda + \Delta\lambda]) \rightarrow 0$ for any m, k . This means ♠ → 0 and thus $\lim_{\Delta\lambda \rightarrow 0} |C(r_{\lambda+\Delta\lambda}^*) - C(r_{\lambda}^*)| = 0$. □

Proposition 8. Suppose $\mathbb{P}(\mathbf{y}|\mathbf{x})$ and $\mathbb{P}(\mathbf{x})$ are continuous in \mathbf{x} . Let $r_0^* \in \arg \min_r L(r, 0)$ where $L(r, \lambda) = S(r) + \lambda \cdot C(r)$ (see also (16)). For any budget $B \in (0, C(r_0^*))$ in the constrained problem in (6) (or equivalently in (15)), there exists $\lambda_{\mathfrak{H}} \geq 0$ such that the minimiser of $L(r, \lambda_{\mathfrak{H}})$ also minimises (6).

Proof. Since $\mathbb{P}(\mathbf{y}|\mathbf{x})$ and $\mathbb{P}(\mathbf{x})$ are continuous in \mathbf{x} , and $\lambda \mapsto C(r_{\lambda}^*)$ is continuous (by Lemma 7), there exists $\lambda_B \geq 0$ such that $C(r_{\lambda_B}^*) = B$. Applying Lemma 4 with this choice of λ_B implies that

$$S(r_{\lambda_B}^*) \leq S(r),$$

for any r such that $C(r) \leq C(r_{\lambda_B}^*) = B$. This proves that $r_{\lambda_B}^*$ is the minimiser of (6). Setting $\lambda_{\mathfrak{H}} = \lambda_B$ completes the proof. □

C.2 Proof of Proposition 1

Proposition (Restated). Assume that $\mathbf{x} \sim \mathbb{P}$ and $\eta(\mathbf{x}) = \mathbb{P}(\mathbf{y} \mid \mathbf{x})$ are continuous random variables. Then, for any LLM meta-distribution \mathfrak{H} , and budget $B \in (0, C(r_0^*))$ (see §C for the definitions of C and r_0), there exists $\lambda_{\mathfrak{H}} \geq 0$ such that the optimal dynamic router r^* for the constrained optimization in (6) is

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

Note that the continuity assumption on the data distribution applies to the setting where \mathbf{x} can be represented via a fixed-length sentence embedding (i.e., a Euclidean vector). This is the primary setting of this work.

Proof. Under the continuity assumption on \mathbb{P} , by Proposition 8, there exists $\lambda_{\mathfrak{H}} \geq 0$ such that the minimiser of the unconstrained objective $L(r, \lambda_{\mathfrak{H}})$ (see (16)) also minimises (6). By Lemma 3, r^* is established. □

C.3 Proof of Proposition 2

Proposition (Restated). For a set of LLMs \mathcal{H} , let r^* denote the Bayes-optimal routing rule in Proposition 1. For any $\mathbf{x} \in \mathcal{X}$ and $h^{(m)} \in \mathcal{H}$, let:

$$\Delta_k(\mathbf{x}, h^{(m)}) = \left| \mathbb{P}_{\mathbf{y}|\mathbf{x}, z=k} [h(\mathbf{x}) \neq \mathbf{y}] - \Psi_k^*(h^{(m)}) \right|.$$

Let $R_{01}(r, \mathcal{H}_{\text{te}}) \doteq \sum_n \mathbb{P} [h_{\text{te}}^{(n)}(\mathbf{x}) \neq \mathbf{y} \wedge r(\mathbf{x}, \mathcal{H}_{\text{te}}) = m]$ denote the 0-1 risk. Then under the regularity condition on \mathbb{P} in Proposition 1, the difference in 0-1 risk between \tilde{r}^* and r^* can be bounded as:

$$\mathbb{E}_{\mathcal{H}_{\text{te}}} [R_{01}(\tilde{r}^*, \mathcal{H}_{\text{te}})] \leq \mathbb{E}_{\mathcal{H}_{\text{te}}} [R_{01}(r^*, \mathcal{H}_{\text{te}})] + \mathbb{E}_{(\mathbf{x}, \mathcal{H}_{\text{te}})} \left[\max_{m \in [|\mathcal{H}_{\text{te}}|], k \in [K]} \Delta_k(\mathbf{x}, h_{\text{te}}^{(m)}) \right].$$

For simplicity, we will refer to \mathcal{H}_{te} simply by \mathcal{H} , and $h_{\text{te}}^{(m)}$ by $h^{(m)}$. We define a proxy risk objective:

$$\tilde{R}_{01}(r, \mathcal{H}) = \mathbb{E}_{\mathbf{x}, z} \left[\sum_{m \in [|\mathcal{H}|]} \Psi_z^*(h^{(m)}) \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right], \quad (18)$$

where the expectation is over the joint distribution over (\mathbf{x}, z) (and not (\mathbf{x}, \mathbf{y})).

Consider solving a variant of the constrained optimization problem in (6) where the original risk objective is replaced with the proxy risk in (18):

$$\begin{aligned} & \min_r \mathbb{E}_{\mathcal{H}} [\tilde{R}_{01}(r, \mathcal{H})] \\ \text{s.t. } & \mathbb{E}_{(\mathbf{x}, \mathbf{y}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} c^{(m)} \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] \leq B. \end{aligned} \quad (19)$$

We can then show that the optimal solution to above proxy constrained optimization problem admits the same form as the cluster-based routing rule \tilde{r}^* in (14):

Lemma C.1. Under the assumption on \mathbb{P} in Proposition 2, for any set of models \mathcal{H} , the minimizer of the proxy constrained optimization problem in (19) is given by:

$$\tilde{r}^*(\mathbf{x}, \mathcal{H}) = \operatorname{argmin}_{m \in [|\mathcal{H}|]} \sum_{k \in [K]} \mathbb{P}(z = k | \mathbf{x}) \cdot \Psi_k^*(h^{(m)}) + \lambda \cdot c^{(m)},$$

for some $\lambda \geq 0$.

We will also find it useful to bound the difference between the original risk $R_{01}(r, \mathcal{H})$ and the proxy risk in (18):

Lemma C.2. For any routing rule r and fixed \mathcal{H} ,

$$\left| R_{01}(r, \mathcal{H}) - \tilde{R}_{01}(r, \mathcal{H}) \right| \leq \mathbb{E}_{\mathbf{x}} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\mathbf{x}, h^{(m)}) \right].$$

We are now ready to prove Proposition 2:

Proof. The excess risk we wish to bound is given by:

$$\begin{aligned} & \mathbb{E}_{\mathcal{H}} [R_{01}(\tilde{r}^*, \mathcal{H}) - R_{01}(r^*, \mathcal{H})] \\ &= \mathbb{E}_{\mathcal{H}} \left[\left\{ R_{01}(\tilde{r}^*, \mathcal{H}) - \tilde{R}_{01}(\tilde{r}^*, \mathcal{H}) \right\} + \tilde{R}_{01}(\tilde{r}^*, \mathcal{H}) - \tilde{R}_{01}(r^*, \mathcal{H}) + \left\{ \tilde{R}_{01}(r^*, \mathcal{H}) - R_{01}(r^*, \mathcal{H}) \right\} \right] \\ &\stackrel{(a)}{\leq} \mathbb{E}_{\mathcal{H}} \left[\tilde{R}_{01}(\tilde{r}^*, \mathcal{H}) \right] - \mathbb{E}_{\mathcal{H}} \left[\tilde{R}_{01}(r^*, \mathcal{H}) \right] + 2 \cdot \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\mathbf{x}, h^{(m)}) \right] \\ &\stackrel{(b)}{\leq} 0 + 2 \cdot \mathbb{E}_{(\mathbf{x}, \mathcal{H})} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\mathbf{x}, h^{(m)}) \right], \end{aligned}$$

as desired. To derive (a), we apply Lemma C.2 to bound the first and third terms. To derive (b), we use the fact that \tilde{r}^* is the minimizer of the proxy-risk $\mathbb{E}_{\mathcal{H}} [\tilde{R}_{01}(\cdot, \mathcal{H})]$ subject to the budget constraint in (19); since r^* also satisfies the same budget constraint, it has an equal or higher expected risk than \tilde{r}^* . \square

We now prove Lemmas C.1 and C.2.

Proof of Lemma C.1. Under the assumption on \mathbb{P} , the constrained problem in (19) is equivalent to minimizing the following Lagrangian objective for some Lagrange multiplier λ [Neyman and Pearson, 1933]:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{(\mathbf{x}, z, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \Psi_z^*(h^{(m)}) \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] + \lambda \cdot \mathbb{E}_{(\mathbf{x}, \mathbf{y}, \mathcal{H})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot c^{(m)} \right] \\ &\stackrel{(a)}{=} \mathbb{E}_{\mathcal{H}} \mathbb{E}_{\mathbf{x}} \mathbb{E}_{z|\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \left\{ \Psi_z^*(h^{(m)}) + \lambda \cdot c^{(m)} \right\} \right] \\ &= \mathbb{E}_{\mathcal{H}} \mathbb{E}_{\mathbf{x}} \underbrace{\left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}(r(\mathbf{x}, \mathcal{H}) = m) \cdot \left\{ \sum_{k \in [K]} \mathbb{P}(z = k | \mathbf{x}) \cdot \Psi_k^*(h^{(m)}) + \lambda \cdot c^{(m)} \right\} \right]}_{\mathcal{L}_{\mathcal{H}, \mathbf{x}}},\end{aligned}$$

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

$$\tilde{r}^*(\mathbf{x}, \mathcal{H}) = \operatorname{argmin}_{m \in [|\mathcal{H}|]} \sum_{k \in [K]} \mathbb{P}(z = k | \mathbf{x}) \cdot \Psi_k^*(h^{(m)}) + \lambda \cdot c^{(m)}.$$

□

Proof of Lemma C.2. Expanding the original risk, we have:

$$\begin{aligned}R_{01}(r, \mathcal{H}) &= \mathbb{E}_{(\mathbf{x}, \mathbf{y})} \left[\sum_{m \in [|\mathcal{H}|]} \mathbf{1}[h^{(m)}(\mathbf{x}) \neq \mathbf{y}] \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \mathbb{E}_{\mathbf{y}|\mathbf{x}} \left[\mathbf{1}[h^{(m)}(\mathbf{x}) \neq \mathbf{y}] \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \mathbb{P}_{\mathbf{y}|\mathbf{x}} \left[h^{(m)}(\mathbf{x}) \neq \mathbf{y} \right] \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \sum_{k \in [K]} \pi_k \cdot \mathbb{P}_{\mathbf{y}|\mathbf{x}, z=k} \left[h^{(m)}(\mathbf{x}) \neq \mathbf{y} \right] \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right]. \quad (20)\end{aligned}$$

Recall that:

$$\begin{aligned}\Delta_k(\mathbf{x}, h^{(m)}) &= \left| \mathbb{P}_{\mathbf{y}|\mathbf{x}, z=k} \left[h^{(m)}(\mathbf{x}) \neq \mathbf{y} \right] - \Psi_k^*(h^{(m)}) \right| \\ &= \left| \mathbb{P}_{\mathbf{y}|\mathbf{x}, z=k} \left[h^{(m)}(\mathbf{x}) \neq \mathbf{y} \right] - \mathbb{P}_{\mathbf{x}', \mathbf{y}'|z=k} \left[h^{(m)}(\mathbf{x}') \neq \mathbf{y}' \right] \right|.\end{aligned}$$

We may next bound (20) in terms of $\Delta_m(\mathbf{x}, h^{(m)})$:

$$\begin{aligned}R_{01}(r, \mathcal{H}) &\leq \mathbb{E}_{\mathbf{x}} \left[\sum_{m \in [|\mathcal{H}|]} \sum_{k \in [K]} \pi_k \cdot \left(\mathbb{P}_{\mathbf{x}', \mathbf{y}'|z=k} \left[h^{(m)}(\mathbf{x}') \neq \mathbf{y}' \right] + \Delta_k(\mathbf{x}, h^{(m)}) \right) \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right] \\ &= \mathbb{E}_{\mathbf{x}} \left[\sum_{k \in [K]} \pi_k \cdot \sum_{m \in [|\mathcal{H}|]} \mathbb{P}_{\mathbf{x}', \mathbf{y}'|z=k} \left[h^{(m)}(\mathbf{x}') \neq \mathbf{y}' \right] \cdot \mathbf{1}[r(\mathbf{x}, \mathcal{H}) = m] \right]\end{aligned}$$

$$\begin{aligned}
& + \mathbb{E}_{\boldsymbol{x}} \left[\sum_{k \in [K]} \pi_k \cdot \sum_{m \in [|\mathcal{H}|]} \Delta_k(\boldsymbol{x}, h^{(m)}) \cdot \mathbf{1}[r(\boldsymbol{x}, \mathcal{H}) = m] \right] \\
& \stackrel{(a)}{\leq} \mathbb{E}_{\boldsymbol{x}} \left[\sum_{k \in [K]} \pi_k \cdot \sum_{m \in [|\mathcal{H}|]} \mathbb{P}_{\boldsymbol{x}', \boldsymbol{y}' | z=k} [h^{(m)}(\boldsymbol{x}') \neq \boldsymbol{y}'] \cdot \mathbf{1}[r(\boldsymbol{x}, \mathcal{H}) = m] \right] + \mathbb{E}_{\boldsymbol{x}} \left[\sum_{k \in [K]} \pi_k \cdot \max_{m \in [|\mathcal{H}|]} \Delta_k(\boldsymbol{x}, h^{(m)}) \right] \\
& \stackrel{(b)}{\leq} \mathbb{E}_{\boldsymbol{x}} \left[\sum_{k \in [K]} \pi_k \cdot \sum_{m \in [|\mathcal{H}|]} \mathbb{P}_{\boldsymbol{x}', \boldsymbol{y}' | z=k} [h^{(m)}(\boldsymbol{x}') \neq \boldsymbol{y}'] \cdot \mathbf{1}[r(\boldsymbol{x}, \mathcal{H}) = m] \right] + \mathbb{E}_{\boldsymbol{x}} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\boldsymbol{x}, h^{(m)}) \right] \\
& = \mathbb{E}_{(\boldsymbol{x}, z)} \left[\sum_{m \in [|\mathcal{H}|]} \Psi_z^*(h^{(m)}) \cdot \mathbf{1}[r(\boldsymbol{x}, \mathcal{H}) = m] \right] + \mathbb{E}_{\boldsymbol{x}} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\boldsymbol{x}, h^{(m)}) \right] \\
& = \tilde{R}_{01}(r, \mathcal{H}) + \mathbb{E}_{\boldsymbol{x}} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\boldsymbol{x}, h^{(m)}) \right].
\end{aligned}$$

where (a) uses the fact that $\sum_m \mathbf{1}[r(\boldsymbol{x}, \mathcal{H}) = m] = 1$ and (b) follows from the fact that $\sum_k \pi_k = 1$.

One can similarly show that:

$$R_{01}(r, \mathcal{H}) \geq \tilde{R}_{01}(r, \mathcal{H}) - \mathbb{E}_{\boldsymbol{x}} \left[\max_{m \in [|\mathcal{H}|], k \in [K]} \Delta_k(\boldsymbol{x}, h^{(m)}) \right],$$

which completes the proof. \square

D Zero Routing

An elementary approach to multi-model routing is to identify the points on the non-decreasing convex hull of the set of cost-risk pairs $\{(c^{(m)}, R(h^{(m)})) : m \in [M]\}$, and to route amongst the corresponding LLMs [Hu et al., 2024b]. Specifically, given a budget $B \in (c^{(1)}, c^{(M)})$, we may pick the two nearest costs $c^{(m_1)} \leq B < c^{(m_2)}$ from the non-decreasing convex hull, and route a query to LLM $h^{(m_1)}$ with probability $\frac{c^{(m_2)} - B}{c^{(m_2)} - c^{(m_1)}}$ and to $h^{(m_2)}$ with probability $\frac{B - c^{(m_1)}}{c^{(m_2)} - c^{(m_1)}}$.

This approach can be seen as a *random router* that randomizes between two LLMs, where the LLMs and mixing coefficients are chosen to maximize the expected quality on the validation sample, while satisfying the budget constraint. Despite its simplicity (i.e., the routing decision being agnostic to the input), this approach was noted as a strong baseline in Hu et al. [2024b].

D.1 Special Case: Optimal Cluster Routing Rule for $K = 1$

When the number of clusters $K = 1$, the routing rule in (9) returns the same LLM for all queries \boldsymbol{x} , and is given by:

$$r(\boldsymbol{x}, \mathcal{H}_{\text{te}}) = \operatorname{argmin}_{n \in [N]} [\Psi(h_{\text{te}}^{(n)}) + \lambda \cdot c(h_{\text{te}}^{(n)})], \quad (21)$$

where $\Psi(h_{\text{te}}^{(n)}) \doteq \frac{1}{N_{\text{val}}} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S_{\text{val}}} \mathbf{1}[h_{\text{te}}^{(n)}(\boldsymbol{x}) \neq \boldsymbol{y}]$. This rule is closely aligned with the Pareto-random router.

Proposition 9. *For any $\lambda \in \mathbb{R}_{\geq 0}$, the routing rule in (21) returns an LLM on the non-decreasing convex hull of the set of cost-risk pairs $\{(c(h_{\text{te}}^{(n)}), r_{01}(h_{\text{te}}^{(n)})) : n \in [N]\}$, where $r_{01}(h_{\text{te}}^{(n)}) = \frac{1}{N_{\text{val}}} \sum_{(\boldsymbol{x}, \boldsymbol{y}) \in S_{\text{val}}} \mathbf{1}[h_{\text{te}}^{(n)}(\boldsymbol{x}) \neq \boldsymbol{y}]$.*

Proof. Suppose there exists a $\lambda_1 \in \mathbb{R}_{\geq 0}$ and $m_1 \in \operatorname{argmin}_m \Psi(h_{\text{te}}^{(m)}) + \lambda_1 \cdot c^{(m)}$ such that $(c^{(m_1)}, r_{01}(h^{(m_1)}))$ is not on the non-decreasing convex hull. Then there exists $h^{(m_2)} \in \mathcal{H}$ such that either $c^{(m_2)} < c^{(m_1)}$ and $r_{01}(h^{(m_2)}) \leq r_{01}(h^{(m_1)})$, or $c^{(m_2)} \leq c^{(m_1)}$ and $r_{01}(h^{(m_2)}) < r_{01}(h^{(m_1)})$. In either case, $\Psi(h_{\text{te}}^{(m_2)}) + \lambda_1 \cdot c^{(m_2)} = r_{01}(h_{\text{te}}^{(m_2)}) + \lambda_1 \cdot c^{(m_2)} < r_{01}(h_{\text{te}}^{(m_1)}) + \lambda_1 \cdot c^{(m_1)} = \Psi(h_{\text{te}}^{(m_1)}) + \lambda_1 \cdot c^{(m_1)}$, which contradicts the fact that $m_1 \in \operatorname{argmin}_m [\Psi(h_{\text{te}}^{(m)}) + \lambda_1 \cdot c^{(m)}]$. \square

E Experimental Setup

We provide more details on the experiments discussed in Section 7.

E.1 Splitting Data and LLMs

In the experiment on each of the four datasets (EmbedLLM [Zhuang et al., 2024], MixInstruct [Jiang et al., 2023], RouterBench [Hu et al., 2024b]), and Math+Code dataset [Dekoninck et al. [2025]]), we split the data into three disjoint portions: train, validation, and test. 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.

Data split Model split	Train	Validation	Test
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 exceptions are the clairvoyant oracle baselines which are allowed access to correctness labels of testing models on training examples. In other words, unlike other baselines, these oracle methods observe all models during training, and are trained on both training and validation portions. These baselines are 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 UniRoute (K -Means) 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.

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 UniRoute (K -Means) approach learns to route among the training models, and is tested on the test set to route among the testing models.

For the Math+Code dataset alone (in Figure 2), we have no training LLMs; so the training sample is unlabeled.

E.2 Evaluation: Deferral Curve

Routing performance may be assessed via a *deferral curve* [Jitkrittum et al., 2023, Wang et al., 2024a, Hu et al., 2024b] $\mathcal{C} = \{(B, R(h_{RM}(\cdot, r_B)) : B \in [c^{(1)}, c^{(M)}]\}$, 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 $R(h_{RM}(\cdot, r_B))$. We may also use a quality metric (e.g., accuracy) in place of the loss to capture quality-cost trade-offs.

E.3 Implementation Details of UniRoute (LearnedMap)

In this section, we give details on the architecture and training of our proposed UniRoute (LearnedMap).

Architecture Recall from Section 5.2 that the LearnedMap attempts to learn $x \mapsto \Phi_{\text{clust}}(x; \theta)$ parameterised by θ . This is a function that maps an input prompt x to a probability vector $\Phi_{\text{clust}}(x; \theta) \in \Delta^K$ where Δ^K denotes the K -dimensional probability simplex. In experiments, $\Phi_{\text{clust}}(\cdot; \theta)$ is defined by an MLP with two hidden layers:

$$\Phi_{\text{clust}}(x; \theta) \doteq (\text{Softmax} \circ \text{FC}(M) \circ H' \circ H \circ \text{BN} \circ \varphi)(x), \quad (22)$$

$$H \doteq [\text{ReLU} \circ \text{BN} \circ \text{FC}(128)], \quad (23)$$

where:

- \circ denotes function composition,
- H and H' denote the two, separate (i.e., no parameter sharing) hidden layers of the same architecture as described in (23),
- $FC(z)$ denotes a fully connected layer with $z \in \mathbb{N}$ outputs,
- $ReLU$ denotes the rectified linear unit i.e., $ReLU(a) \doteq \max(0, a)$,
- $Softmax$ denotes a softmax layer,
- BN denotes the batch normalization layer, and
- φ denotes a frozen text embedding.

Recall from the experiments in Section 7 that we use Gecko Lee et al. [2024b] for the text embedding φ .

Training We use Keras for implementing (22). In all experiments on EmbedLLM, MixInstruct, and RouterBench, the Learned Cluster Map is trained for only 5 epochs using Adam as the optimization algorithm. We observe that training for too long can lead to overfitting. Training batch size is set to 64, and the learning rate is 0.005. Since φ is frozen, training $\Phi_{\text{clust}}(\cdot; \theta)$ amounts to training the MLP with two hidden layers. It is sufficient to use CPUs for training. For one trial, training takes only a few minutes to complete.

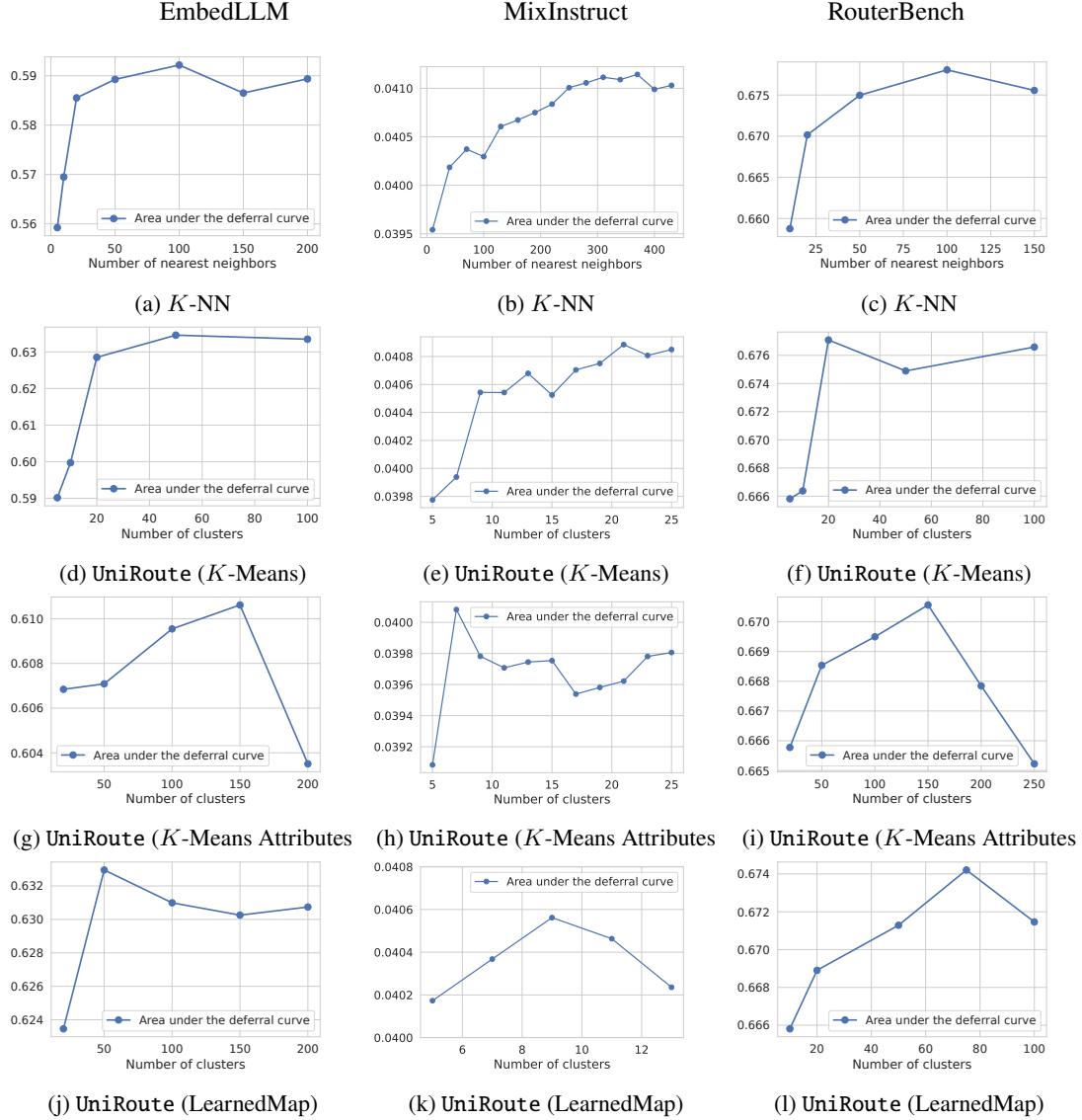


Figure 4: Validation performance of four methods considered in Figure 2 and Appendix F.3: *K*-NN, UniRoute (*K*-Means), UniRoute (*K*-Means Attributes), and UniRoute (LearnedMap). See Appendix F.1 for more details.

E.4 Router Cost

All routing methods rely on a frozen prompt embedding model φ . As described in Appendix F.2, φ is set to be the token probability quantiles of the Gemma 2B model in the Headlines dataset. For other datasets, we use Gecko 1B model [Lee et al., 2024b] for φ . Importantly, all methods rely on the same embedding model, and thus share the same overhead for prompt embedding.

At inference time, for UniRoute (*K*-Means), deciding the LLM to route to for a given test query involves computing the text embedding, and finding the nearest centroid out of K centroids (recall that K -means is run only once on the training set). For UniRoute (LearnedMap), after computing the prompt embedding, we pass the embedding through a small MLP as described in (22). Compared to the sizes of candidate LLMs (up to 60+B in EmbedLLM, for instance), invoking the text embedding model and performing a few operations after (centroid lookup, or invoking a small MLP) incur a negligible cost.

When plotting the deferral curves, we do not include the cost of the routing model. Including the router’s overhead will simply shift all the deferral curves to the right by a small amount, without changing the relative ordering of the methods.

F Additional Experimental Results

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

F.1 Hyper-parameter Choices & Statistical Significance

There are three methods that we consider in the experiments in Section 7 that depend on a hyperparameter K . Specifically, K in K-NN refers to the number of nearest neighbors, and K in UniRoute (K -means) and UniRoute (LearnedMap) refers to the number of clusters. In Figure 2, we report the performance of these methods with the best K found on each dataset separately. We now describe the validation procedure we used to select the best K .

K-NN For each candidate K , and each prompt in the validation set, find the K nearest queries in the training set (in the Gecko embedding space). Route each prompt in the validation set to the most appropriate *training LLM* according to the routing rule (8) where $\gamma^{(m)}$ is estimated with (5). Produce a deferral curve on the validation set, and compute the normalized area under such curve. Select K that maximizes the area. The list of candidate K ’s is set to be from 5 to one third of the validation sample size.

UniRoute (K -Means) For each candidate K , perform K -means on the training set using Gecko embeddings [Lee et al., 2024b]. Compute the feature vector representation of each *training LLM* on the training set using (13). For each prompt in the validation set, find the nearest cluster, and route the prompt to the most appropriate *training LLM* according to the routing rule (9). Produce a deferral curve on the validation set, and compute the normalized area under such curve. Select K that maximizes the area. The list of candidate K ’s is from 3 to the number of validation sample size, divided by 50.

UniRoute (K -Means Attributes) An alternate approach that we consider in Appendix F.3 is to construct a binary vector of *prompt attributes*, denoting whether a prompt possesses certain characteristics [Li et al., 2024b,c], e.g., whether it requires multi-step reasoning, seeks a single correct answer, and so on. These can be seen as a generalised “task”. Compared to a general purpose text embedding, such a representation is a coarser representation; on the other hand, for the purposes of model routing, this can help mitigate overfitting.

Concretely, we parameterize the prompt embedding model to be $\Phi(\mathbf{x}) = \sigma(\mathbf{V}^\top \varphi_{\text{Gecko}}(\mathbf{x}))$ 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 φ_{Gecko} frozen) by minimizing the sigmoid cross entropy to predict whether the j -th semantic attribute is active on each input prompt. We use the seven prompt difficulty attributes as described in Li et al. [2024b], and prompt Gemini 1.5 Pro 002 to annotate each binary attribute on each training example. Once the prompt 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 .

UniRoute (LearnedMap) For each candidate K , perform K -means on the training set using Gecko embeddings. Compute the feature vector representation of each *training LLM* on the training set using (13). Parameterize the cluster assignment map with a softmax-dense layer as described in Section 5.2, resulting in a parameter θ to learn. Learn the parameter by minimizing the binary sigmoid cross entropy on the training with the labels given by the correctness labels of the *training LLMs*. With the cluster map trained, for each prompt in the validation set, route the prompt to the most appropriate training LLM according to the routing rule (9). Produce a deferral curve on the validation set, and compute the normalized area under such curve. Select K that maximizes the area. The list of candidate K ’s is from 3 to the number of validation sample size, divided by 50.

The above tuning procedure for K cannot be applied to the Math+Code dataset considered in Section 7, where we have no training LLMs. In this case, we set K to $\sqrt{N_{\text{val}}}$ for K-NN and to $N_{\text{val}}/50$ for K -Means.

Effect of K Figure 4 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.

MLP (Clairvoyant) For this oracle baseline, as per [Hu et al., 2024b], we fix the number of hidden layers to two and the number of hidden nodes in each hidden layer to 100. We fit the MLP on the combined training and validation set to predict a quality score for each test LLM, and route via (2). For training, we once again ran 30 epochs of Adam with a learning rate of 10^{-3} and batch size 64, and picked the checkpoint that yielded the best quality metric on the validation set.

Matrix Factorization (Clairvoyant) For this oracle baseline, we fit a factorized model $\mathbf{A} \cdot \mathbf{B}$, with $\mathbf{A} \in \mathbb{R}^{D_p \times d}$, $\mathbf{B} \in \mathbb{R}^{d \times m}$, where D_p is the dimension of the pretrained query embedding, d is an intermediate dimension, and N is the number of test LLMs to route to. We choose d using the same procedure used to pick the hyper-parameter K above. For a query $x \in \mathcal{X}$ and pre-trained $\varphi(x) \in \mathbb{R}^{D_p}$, this router predicts N LLM scores via $\mathbf{A} \cdot \mathbf{B} \cdot \varphi(x)$, and routes via (2). For training, we ran 30 epochs of Adam with a learning rate of 10^{-3} and batch size 64, and picked the checkpoint that yielded the best quality metric on the validation set.

Statistical Significance For the table in Figure 2 (Top), we repeat the experiments over 400 trials, and report statistical significance (via the sign test) at significance level $\alpha = 0.01$. For the plots of area vs. validation sample size in Figure 2 (Bottom), we repeat the experiments over 100 trials for EmbedLLM, RouterBench, SPROUT o3-mini and 1000 trials for Math+Code, and report 96% confidence intervals (i.e. two-sigma standard errors).

F.2 Deferral Curves and Additional Comparisons

As described in Section 7, for each of the 400 independent trials, we randomly split examples of each dataset into training, validation and testing. Figure 5 presents the mean deferral curves of all the methods we consider. All the statistics (Area (50%), Area and QNC) reported in Figure 2 are derived from these curves. All results here are for the dynamic LLM pool setting as considered in Figure 2.

EmbedLLM Figure 5a presents deferral curves on the EmbedLLM problem. These results are the same as in Figure 3 where we add two oracle baselines: Matrix Factorization (MatFac) [Ong et al., 2025, Zhuang et al., 2024], and MLP [Hu et al., 2024b]. We reemphasize that these two baselines are not designed for the dynamic pool setting, and are allowed to observe test LLMs during training to be applicable to our setting.

Headlines We consider the Headlines dataset as used in Chen et al. [2023b] consisting of 10K prompts in total. Each prompt asks an LLM to determine the price direction (up, down, neutral, or none) of an item in a list of news headlines. There are 12 LLMs of various sizes in this dataset. Each LLM has a distinct cost (USD) of processing one prompt, which is a function of the input length, output length, and a fixed cost per request (see Table 1 in Chen et al. [2023b]). We hold out six LLMs for training and the other six LLMs for testing:

- Training LLMs: Textsynth FAIRSEQ, OpenAI GPT-4, OpenAI GPT-Curie, Textsynth GPT-Neox, AI21 J1-Large, Cohere Xlarge;
- Test LLMs: OpenAI ChatGPT, OpenAI GPT-3, Textsynth GPT-J, AI21 J1-Grande, AI21 J1-Jumbo, Cohere Medium.

In each of the 400 independent trials, we randomly partition the data into 4000, 400, 5600 examples for training, validation, and testing, respectively. For Headlines, unlike other datasets, we do *not* use Gecko for embedding prompts. Rather, the frozen text embedding φ is constructed based on the output of the Gemma 2B model¹: for each prompt, its embedding is the seven equally spaced quantiles of the per-token probabilities of the output tokens from Gemma 2B.

The mean deferral curves over 400 trials are shown in Figure 5d. We observe that our proposed UniRoute (LearnedMap) has higher accuracy than K -NN throughout the whole cost range.

MixInstruct MixInstruct is a dataset from Jiang et al. [2023] containing responses from 11 LLMs. We use a random split of 50% of the LLMs for training and the rest for testing. We use (exponentiated) BARTScore [Yuan et al., 2021] as the evaluation metric, following Jiang et al. [2023]. The MixInstruct dataset does not have LLM API costs annotated. We use the number of parameters of the LLM as the cost of processing one prompt.

¹Gemma 2B model: <https://huggingface.co/google/gemma-2b>.

Method	Dataset	EmbedLLM			RouterBench			SPROUT o3-mini			Headlines			MixInstruct		
		Area (50%) ↑	Area ↑	QNC ↓	Area (50%) ↑	Area ↑	QNC ↓	Area (50%) ↑	Area ↑	QNC ↓	Area (50%) ↑	Area ↑	QNC ↓	Area (50%) ↑	Area ↑	QNC ↓
ZeroRouter		.285 *	.607 *	87.5% *	.320 *	.689 *	99.9%	.404 *	.820 *	100.0% *	.0233 *	.0487 *	96.8%	.380 *	.819 *	88.0% *
K-NN		.298 *	.636 *	46.1% *	.328 *	.707 *	99.7%	.418 *	.844 *	29.6% *	.0235 *	.0494	94.8%	.411	.830 *	43.7% *
UniRoute (<i>K</i> -Means)		.307 *	.648 *	33.9%	.332	.712	99.4%	.421	.850	19.6%	.0235 *	.0491	95.2%	.409 *	.828 *	56.9% *
UniRoute (LearnedMap)		.308	.651	33.2%	.331	.711	99.6%	.420	.846	23.4%	.0236	.0491	96.2%	.411	.832	34.9%
MLP (Clairvoyant)		.314	.664	26.9%	.339	.723	95.2%	.427	.859	4.5%	.0240	.0502	85.7%	.412	.835	18.1%
Matrix Fac. (Clairvoyant)		.313	.663	27.7%	.338	.720	99.5%	.426	.857	5.0%	.0242	.0505	82.9%	.413	.835	13.7%

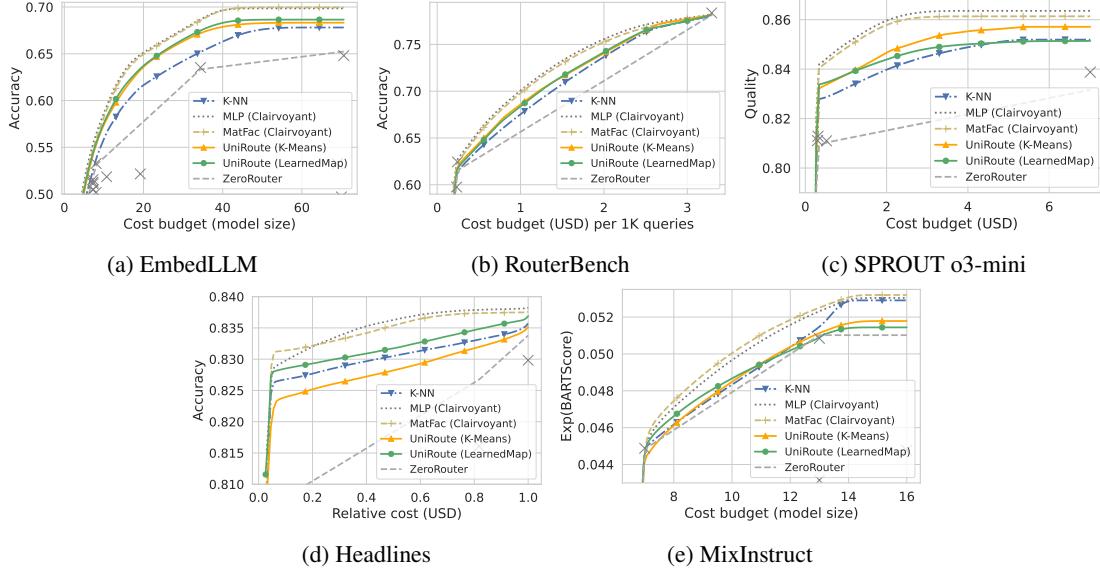


Figure 5: Deferral curves and router evaluation metrics (Area (50%), Area, and QNC) for different methods in the dynamic pool setting. MLP [Hu et al., 2024a] and MatFac [Ong et al., 2025, Zhuang et al., 2024], are oracle methods that observe testing LLMs during training. ZeroRouter [Hu et al., 2024b] and *K*-NN [Hu et al., 2024b], Shnitzer et al. [2023] are baselines applicable to the dynamic LLM pool setting.

F.3 Train on Chatbot Arena and Test on EmbedLLM

We observe that representing each prompt with a small number of binary attributes that capture its inherent hardness shines when there is a prompt distribution shift at test time. To illustrate this, we use the seven binary difficulty attributes proposed in Li et al. [2024b], and prompt Gemini 1.5 Pro to annotate each attribute for each training prompt. We then construct a query embedder

$$\varphi(\mathbf{x}) = \sigma(\mathbf{V}^\top \text{Gecko}(\mathbf{x})) \in [0, 1]^7, \quad (24)$$

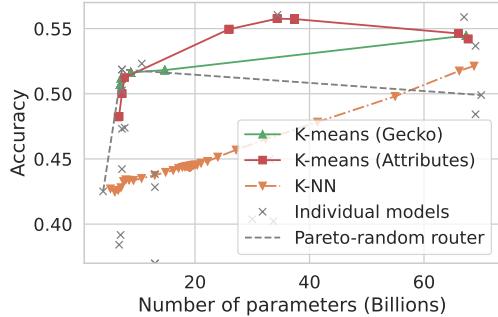
where $\mathbf{V} \in \mathbb{R}^{768 \times 7}$ is distilled using the training set to predict the 7-category attributes for any new prompt \mathbf{x} .

We compare Gecko-based prompt 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 prompt 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 and 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 6 where we compare two variants of our proposed UniRoute (*K*-means):

1. *K*-means (Gecko). This is UniRoute (*K*-means) (see Section 5.1) where the text embedding is set to the Gecko model [Lee et al., 2024b].
2. *K*-means (Attributes). This is UniRoute (*K*-means) where the text embedding is set to (24).



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

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 prompt 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.

F.4 Static LLM Pool Setting

While the dynamic pool setting is the focal point of our work, we show in Table 2 that even in the *static* LLM pool setting, UniRoute is typically comparable to most baselines. The MLP baseline alone often has a slight edge. This is because, unlike our methods, which are tied to a particular input LLM representation, the MLP approach has the flexibility of learning its own representation for each fixed LLM. Note that this is possible only in the static LLM pool setting.

In this case, all LLMs are seen during training, and hence the training and validation samples have correctness label annotations from all LLMs. We tune the hyper-parameters, such as K in K -NN and K -Means, the number of hidden nodes in MLP, and the intermediate dimension in Matrix Factorization, to maximize the area under the deferral curve on the validation sample. We pick K from the range $\{5, 10, 25, 50, 75, 100, 150, 200, 250, 300\}$ for K -NN and K -Means, pruning out values that are too large for the given validation sample size.

For UniRoute (LearnedMap), we used two hidden layers, with the same chosen number of hidden nodes as MLP. We set the number of clusters to be the same as the chosen number of clusters for UniRoute (K-Means). We employed Adam with learning rate 0.001 and batch size 64 for training, picking the checkpoint with the best quality metric on the validation set. For MixInstruct alone, given that the dataset is prone to overfitting, we replicate the same hyper-parameter choices as in the dynamic LLM pool setting (Appendix E.3).

For these experiments, we additionally consider the Headlines dataset (see Appendix F.2) where all LLMs are fully observed during training. We additionally consider the LiveCodeBench coding benchmark [Jain et al., 2024], and include from it 15 LLMs for which the model size was publicly available (the Math+Code dataset contains two LLMs from this benchmark). We split the LiveCodeBench dataset into 60% for training, 10% for validation, and 30% for testing, and employ the same hyper-parameter choices as in MixInstruct. For Headlines, we use 40%, 10%, 50% for training, validation, and testing, respectively.

Table 2: Comparing UniRoute with existing methods for the **static LLM pool** setting, where $\mathcal{H}_{\text{tr}} = \mathcal{H}_{\text{te}}$. We report the area under the deferral curve (\uparrow). The best baseline results are highlighted, and the best UniRoute results are **boldfaced**. Even in the static setting, our approach is competitive compared to most baselines. The MLP baseline often has a slight edge. This is because, unlike our methods, which are tied to a particular input LLM representation, the MLP approach has the flexibility of learning its own representation for each fixed LLM. Note that this is possible only in the static LLM pool setting, and not the dynamic setting, which is the focus of this paper.

Method \ Dataset	EmbedLLM	MixInstruct	RouterBench	Math+Code	LiveCodeBench	Headlines
ZeroRouter [Hu et al., 2024b]	.601	.0483	.707	.392	.457	.834
MLP Hu et al. [2024b]	.689	.0500	.747	.483	.480	.852
Matrix Factorization [Ong et al., 2025, Zhuang et al., 2024]	.682	.0503	.744	.482	.482	.849
<i>K</i> -NN Hu et al. [2024b], Shnitzer et al. [2023]	.636	.0510	.744	.475	.474	.854
UniRoute (<i>K</i> -Means)	.682	.0504	.744	.480	.481	.845
UniRoute (LearnedMap)	.683	.0502	.744	.463	.479	.854

F.5 Visualisation of LLM Embeddings

We visualise the cluster-based LLM embeddings Ψ learned from our clustering procedure on the EmbedLLM and RouterBench datasets in Figures 7 and 8. These heatmaps illustrate the Gaussian kernel similarity between all pairs of LLM embeddings on these datasets. The results are largely intuitive: e.g., on RouterBench, we find that the `claude` family of models are highly similar to each other. Similarly, on EmbedLLM, code-focussed models generally tend to demonstrate a high degree of similarity.

We emphasise again that EmbedLLM previously also considered representing LLMs as feature vectors. Importantly, however, their representation depends on a *fixed* pool of LLMs, and does not readily generalise to new LLMs (without further retraining).

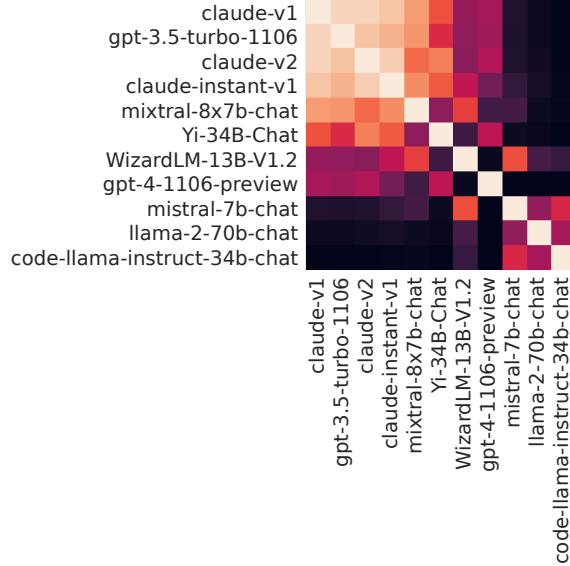


Figure 8: Gaussian kernel similarity between pairs of LLM embeddings on RouterBench dataset.

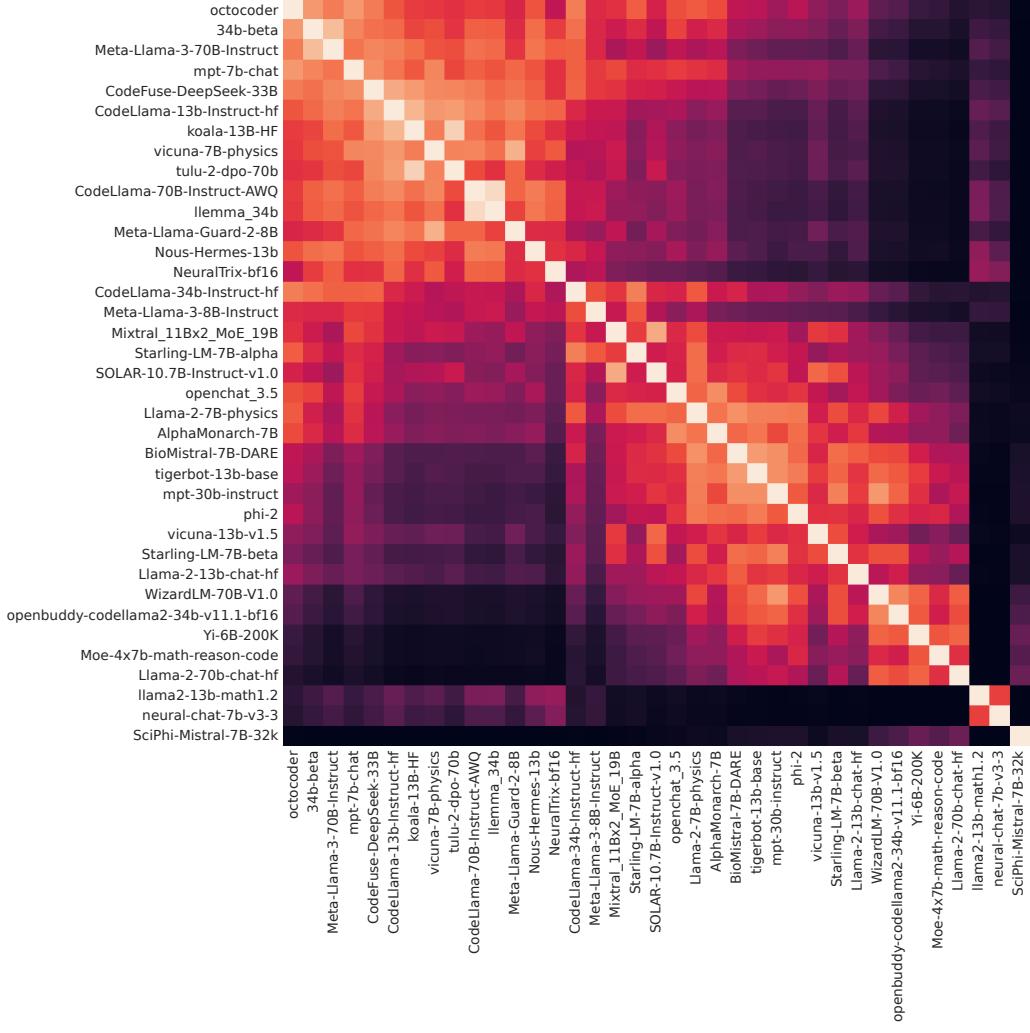


Figure 7: Gaussian kernel similarity between pairs of LLM embeddings on EmbedLLM dataset.