Generalising LLM Routing using Past Performance Retrieval: A Few-Shot Router is Sufficient

Clovis Varangot-Reille[1,2], **Christophe Bouvard**[1], **Antoine Gourru**[2]
[1] Wikit, Lyon, France; [2] Laboratoire Hubert Curien, Université Jean Monnet Saint-Etienne, France
clovis.varangot@wikit.ai, christophe.bouvard@wikit.ai, antoine.gourru@univ-st-etienne.fr

Abstract

We study model routing for Large Language Model (LLM)-based systems. A model, called the router, dynamically chooses which LLM should handle a given input/query. We challenge the assumption that complex routers are necessary for generalising to new candidate LLMs. We introduce CONTEXTUALROUTER, a simple meta-evaluation framework that predicts per-model performance for new queries by retrieving similar past queries and reweighting model scores with lightweight attention. During inference, the router balances estimated performance and cost by adjusting a tunable cost penalty parameter. This allows the router to adapt dynamically to the addition or removal of LLMs without the need for retraining. Across five routing benchmarks (*SPROUT, RouterBench, LiveBench, BigGenBench, and EmbedLLM*), CONTEXTUALROUTER matches the quality–cost trade-offs of other generalisable routers. Surprisingly, a simpler non-parametric baseline, k-nearest-neighbour averaging, performs comparably or better, achieving strong performance estimation, high NDCG, and substantial cost savings. Retrieval-based routers remain robust to k, embedding size, data sparsity, retrieval degradation, and generalise to unseen queries and models with as little as 1% historical data. These results suggest that effective retrieval alone enables generalisable LLM routing.

1 Introduction

Large Language Models (LLM) have been applied in various domains, ranging from conversational agents (Dam et al., 2024) to multi-agent systems (Tran et al., 2025). These systems usually rely on a single LLM to perform the tasks. This model-centric approach focuses on selecting the most suitable LLM for deployment. However, domain specificity and task complexity often differ across user requirements. Therefore, instead of a universal model suitable for all scenarios, each user query may require a distinct model to balance performance and computational cost.

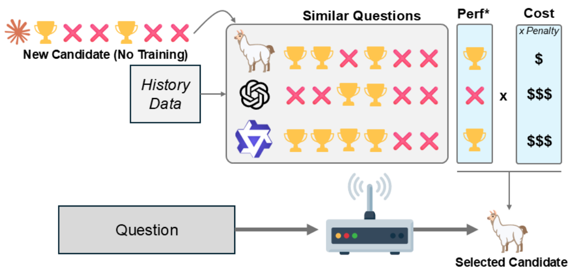


Figure 1: CONTEXTUALROUTER uses historical performance data from a small dataset to predict candidate performance on new queries. At inference, it routes to the candidate that optimizes the trade-off between predicted performance and cost. New candidates can be added without retraining by evaluating them on the history dataset. *Perf*: Estimated performance.*

The routing framework in LLM-based systems, as proposed in earlier works (Hu et al., 2024a; Ong et al., 2025), was proposed as a solution to this issue, aiming to create dynamic systems in which an LLM is selected based on the query. Most existing methods rely on supervised learning, using classifiers for topic selection or regressors for static performance estimation (Jain et al., 2024; Wang et al., 2024; Dekoninck et al., 2025; Ding et al., 2024; Hu et al., 2024b; Somerstep et al., 2025; Liu et al., 2024; Ong et al., 2025; Zhuang et al., 2025). A major limitation of these methods is their inability to generalise to new models without retraining. Recent studies have addressed this challenge through model representation learning, which projects candidate models into latent performance spaces derived from historical data, enabling zero-shot generalisation to unseen models (Jitkrittum et al., 2026; Feng et al., 2025a).

While these representation-based approaches demonstrate strong generalisation capabilities, they often introduce additional architectural complexity through graph neural networks or pseudo–zero-shot designs. This raises a fundamental question: is such sophistication necessary, or can simpler generalisable methods achieve comparable performance without retraining?

In this work, we investigate whether low-resource generalisable routing systems are efficient routing architectures (Figure 1). Methods such as attention-like mechanisms, clustering-based strategies, or averaging over similar samples can achieve comparable performance while reducing inference cost, compared to selecting the best overall or most expensive model. All architectures can generalise to new routing candidates without retraining by representing models in a shared performance space. We show that a simple *k*-NN approach is sufficient, provided that retrieval is efficient, and demonstrate that on several routing evaluation benchmark and different embedding models.

2 Related Works

2.1 Routing in LLM-based Systems

The paradigm of model selection has become increasingly important in the context of LLM. Most current systems are monolithic, relying on a single, generalist LLM (e.g., *Claude-Sonnet*). However, this approach may not always be ideal. Depending on the complexity of the query or the knowledge required, a single model may lack specific ability to provide adequate responses or be overly complex. To address this limitation, several routing strategies have been proposed (Varangot-Reille et al., 2025). Post-generation strategies (i.e., cascade routing) first generate an answer and then evaluate its quality with a scoring function; if the quality is insufficient, the query is routed to another model from a predefined sequence. In contrast, pre-generation routing strategies aim to predict which model is most likely to produce the optimal response before generation. While this approach reduces latency and computational cost, it introduces uncertainty as the model's performance is inferred without observing the output. Most routing strategies in the literature rely on supervised learning, either by training classifiers to decide whether to route a query to a given model (Jain et al., 2024; Wang et al., 2024; Dekoninck et al., 2025; Ding et al., 2024; Jeong et al., 2024; Malekpour et al., 2024; Shnitzer et al., 2024; Stripelis et al., 2024; Srivatsa et al., 2024; Ong et al., 2025), or by training regressors to estimate performance scores (Hu et al., 2024b; Somerstep et al., 2025; Liu et al., 2024; Ong et al., 2025; Zhuang et al., 2025).

2.2 Generalisable Routing in Dynamic Environments via Model Representation

Most of the routing approaches discussed previously lack the ability to adapt to new pooling of routing candidates at inference time. These static pipelines require retraining to accommodate new environments (i.e., different sets of routing candidates). However, this adaptability represents a critical requirement for real-world implementation, as new models and processing strategies emerge continuously. Recent work has proposed architectures designed to address this limitation by projecting routing candidates into a latent representation derived from their historical performance (Feng et al., 2025a; Jitkrittum et al., 2026). In this framework, the router learns to select models based on these latent representations (Feng et al., 2025a; Jitkrittum et al., 2026). Consequently, when implementing a new routing candidate, there is no need to retrain the router; instead, it is only necessary to project the candidate into the existing latent model space.

3 ContextualRouter

3.1 Problem Formulation

Let $\mathcal{Q} = \{q_1, q_2, ..., q_x\}$, a set of x input queries, and $\mathcal{M} = \{M_1, M_2, ...M_n\}$, a set of n LLM candidates for the router. The problem is to optimally assign each query to exactly one model while balancing performance against resource constraints. The performance of a routing candidate M_i on query q_j is captured by the true performance matrix $P = [p_{ij}] \in \mathbb{R}^{n \times x}$, where higher values indicate better performance according to some specific metric (e.g., accuracy, correctness score, etc.).

The basic routing problem aims to find the optimal routing candidate M^* which maximise performance p_{M^*}:

$$M^*(q_j) = \arg\max_{M_i \in \mathcal{M}} p_{ij} \tag{1}$$

3.2 Routing for Resource Optimisation

In a resource optimisation context, the objective extends beyond finding the most performant model for a specific query. Each model m_i has an associated cost $c_i \in \mathbb{R}+$ (e.g., dollar per token, latency), stacked in the cost vector $\mathbf{c} = [c_1, c_2, \ldots, c_n]$. The cost c_q of a query must satisfy $c_q \leq B_q$, where $B_q \in \mathbb{R}+$ is the user budget. We introduce the cost penalty parameter $\lambda \geq 0$, which quantifies the trade-off weight between performance and cost, akin to Hu et al. (2024a)'s performance score or Jitkrittum et al. (2026)'s correctness representation. λ represents the cost one is willing to pay for a one-unit increase in predicted performance. The problem becomes:

$$M^*(q_j) = \arg\max_{M_i \in \mathcal{M}} (p_{ij} - \lambda c_i) \quad (2)$$

where λ, the cost penalty, determines how heavily expensive models are penalized in the selection process. Increasing the value of λ biases the routing towards a cheaper LLM candidate.

While the cost of a routing candidate can be estimated *a priori* (e.g., price per token) and the cost penalty parameter λ is adjustable at inference time, the performance p_{ij} of model M_i on query q_j remains unknown until inference. Thus, the routing optimisation problem requires learning a performance estimation function. We define a parameterised function that maps queries to estimated performance scores:

$$\hat{p}_{ij} = f_\theta(q_j, M_i) \quad (3)$$

where $\hat{p}_{ij}$ represents the estimated performance of model M_i on query q_j, θ denotes the learnable parameters, and $f_\theta : \mathcal{Q} \times \mathcal{M}$ is the performance prediction function. Given the performance estimates, the optimal model selection problem becomes:

$$\begin{aligned} M^*(q_j) &= \arg\max_{M_i \in \mathcal{M}} (\hat{p}_{ij} - \lambda c_i) \\ \hat{p}_{ij} &= f_\theta(q_j, M_i) \end{aligned} \quad (4)$$

3.2.1 Routing Generalisation

Finally, in real-world scenarios, the set of routing candidates $\mathcal{M}_T$ used during training may differ from the set of routing candidates $\mathcal{M}_I$ available at inference time (Jitkrittum et al., 2026; Feng et al., 2025a; Tailor et al., 2024). When only some or none of the training-time candidates are available at inference time, the routing function f_θ cannot rely on the candidate-specific behaviours learned during training. Similarly, the query distribution might shifts from training ($\mathcal{Q}_T$) to inference ($\mathcal{Q}_I$). A generalisable router must rely solely on invariant meta features and avoid conditioning on specific models ($\mathcal{M}$) or specific queries ($\mathcal{Q}$).

3.3 Our Architecture

CONTEXTUALROUTER is based on a contextual meta-evaluation algorithm to predict $\hat{p}_{ij}$ of routing candidate M_i on query q_j. Meta-learning can be defined as *learning to learn* where a model is trained over multiple tasks to learn a general learning function which can generalise across tasks (Vanschoren, 2018). Generalisable routing through model representation can be understood as a form of meta-evaluation. This involves learning to predict how models might perform when faced with queries they have not encountered before, based on previous experience. In other words, through the meta-evaluation process, the router must learn how to route effectively in different, or even unseen, contexts across various routing tasks (e.g., different number of routing candidates) (Vanschoren, 2018). The rationale underlying CONTEXTUALROUTER is that the true performance p_{ij} can be estimated from the observed performances of candidate M_i on a set of queries similar to q_j, collectively referred to as the context $\mathcal{E}(q_j, \mathcal{M})$ for a specific query q_j. Then, the performance estimation function f_θ uses the context to estimate $\hat{p}_{ij}$. Formally, the problem can be reformulated as:

$$\begin{aligned} M^*(q_j) &= \arg\max_{M_i \in \mathcal{M}} (\hat{p}_{ij} - \lambda c_i) \\ \hat{p}_{ij} &= f_\theta(\mathcal{E}(q_j, M_i)) \end{aligned} \quad (5)$$

Linking it to meta-learning taxonomy, the performance matrix and similarity vectors derived from historical data, or $\mathcal{E}(q_j, \mathcal{M})$, can be understood as *meta-features*, a characterization of the current user query and routing context (Vanschoren, 2018). Thus, the CONTEXTUALROUTER works as a *meta-model* that recommends the best configuration, or the most optimal model, given the current task (i.e., query and candidates) (Vanschoren, 2018).

3.3.1 Context Extraction

Each query q is first embedded into a semantic space using an encoder ϕ, i.e., $\phi(q) \in \mathbb{R}^d$. For a given query q_j, we retrieve its k nearest neighbours from the training dataset $\mathcal{D}_T = \{q_1^T, q_2^T, \ldots, q_n^T\}$ by computing the cosine similarity between q_j and each training query q_n^T. We then select the top-k queries with the highest similarity scores as the nearest neighbours of q_j.

The k highest similarity scores form a similarity vector $s \in \mathbb{R}^k$. Using these k neighbours, we construct a neighbour performance matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{M}| \times k}$ where each entry $\mathbf{P}_{ik}$ corresponds to the observed performance of candidate M_i on the k-th neighbour. The context $\mathcal{E}(q_j, \mathcal{M})$ is then defined as the tuple $\mathcal{E}(q_j, \mathcal{M}) = (s, \mathbf{P})$.

3.3.2 Performance Estimation

The performance estimation is formulated as a regression problem to predict the performance score of routing candidates. The input dimensionality is k, the number of nearest neighbours, which ensures that the estimator remains agnostic to the total number of routing candidates $|\mathcal{M}|$.

We introduce an attention-based weighting mechanism to modulate the influence of unreliable performance from distant neighbours. We define an attention-like matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{M}|\times k}$ that reweights candidate performances based on their similarity–performance interactions

Projection into Shared Space. We first project the similarity vector $\mathbf{s} \in \mathbb{R}^k$ and the performance matrix $\mathbf{P} \in \mathbb{R}^{|\mathcal{M}|\times k}$ into a shared latent space of dimension h:

$$\mathbf{S} = W_s\mathbf{s} + b_s \qquad \mathbf{P}' = W_p\mathbf{P} + b_p\mathbf{1}_{|\mathcal{M}|}^\top \tag{6}$$

where $W_s \in \mathbb{R}^{h\times k}$ and $W_p \in \mathbb{R}^{h\times k}$ are learnable projection matrices, and $b_s, b_p \in \mathbb{R}^h$ are bias terms.

Attention Weights Computation. First, the attention weights are computed as

$$\mathbf{A}_\omega = \mathbf{S}\cdot\mathbf{P}'^\top, \qquad \mathbf{A}_\omega \in \mathbb{R}^{k\times|\mathcal{M}|} \tag{7}$$

Then, the softmax function is applied row-wise to obtain the normalised attention weights:

$$\mathbf{A} = \sigma(\mathbf{A}_\omega^\top), \qquad \mathbf{A} \in \mathbb{R}^{|\mathcal{M}|\times k} \tag{8}$$

Each row of $\mathbf{A}$ assigns a normalized relevance score to the performance on the k nearest neighbours. These weights highlight which neighbours contribute most to the predicted performance of each routing candidate, effectively filtering out less informative performances.

Weighted Performance Representation. The attention matrix is then used to weight performance scores element-wise:

$$\mathbf{P}_{\text{att}} = \mathbf{A}\odot\mathbf{P} \tag{9}$$

Feed-Forward Regression Head. We feed the attention-weighted representation into a two-layer feed-forward network with ReLU activations to produce a hidden representation. This hidden representation is then passed through a regression layer to predict the final performance $\hat{p}_{ij}$. To ensure that the performance lies within the $[0,1]$ range, $\hat{p}_{ij}$ is clamped to this interval.

Learning objective. The routing task can be subdivided into two subtasks: (i) estimating the performance of each candidate; (ii) obtaining a ranking that is representative of the final order of the candidates. The first learning objective $\mathcal{L}_{MSE}$ minimizes the mean squared error (MSE) between the predicted and true performance scores over all query–candidate pairs:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{Q}||\mathcal{M}|}\sum_{i=1}^{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{M}|}\left(p_{ij} - \hat{p}_{ij}\right)^2 \tag{10}$$

The second one $\mathcal{L}_{NDCG2++}$ is the NDCG-Loss2++ proposed by Wang et al. (2018). The loss is built to reward if the inferred ranking is close to the true ranking. It is included because the consequences of incorrectly estimating the performance of a high-ranking model must be more severe than those of misestimating a low-ranking one.

Thus, the final loss can be formulated as:

$$\mathcal{L}_{CRout} = \mathcal{L}_{MSE} + \mathcal{L}_{NDCG2++} \tag{11}$$

Routing Candidate Selection. At test time, the estimator outputs the predicted performance scores for all routing candidates $M_i \in \mathcal{M}$ given query q_j:

$$\hat{\boldsymbol{p}}_j = \left[\hat{p}_{1j}, \hat{p}_{2j}, \ldots, \hat{p}_{|\mathcal{M}|j}\right] \in \mathbb{R}^{|\mathcal{M}|} \tag{12}$$

This is analogous to the correctness vector representation proposed by Jitkrittum et al. (2026), except that their method infers the representation from the average performance on the centroids, whereas ContextualRouter derives it from the performance on the top-k nearest neighbours.

This performance prediction vector $\hat{\boldsymbol{p}}_j$ is then passed to Eq.(5) to route to the optimal routing candidate $M^*(q_j)$. The cost penalty is kept independent of the learning process, allowing it to be parametrised at deployment according to application-specific resource requirements without retraining the estimator.

4 Experiment

4.1 Datasets and Metrics

4.1.1 Datasets

The datasets used in our experiments include *SPROUT* (Somerstep et al., 2025), *EmbedLLM* (Zhuang et al., 2025), *RouterBench* (Hu et al., 2024a), *LiveBench Leaderboard* (White et al., 2025) and *BigGenBench Leaderboard* (Kim et al., 2025).We removed Chinese datasets from *Router-Bench* (Hu et al., 2024a) to avoid any language influence that may confound the routing task. Each dataset provides an instruction or question, alongside the ground truth and performance scores for each LLM candidate. These scores represent model performance on the question (See Appendix A for more details on the settings). Performance is evaluated via LLM-as-a-Judge.

4.1.2 Metrics

We evaluated the following metrics: **(i) Peak Performance**: The maximum average true performance of the routing candidates selected by the router. Routing candidates performances are normalised to the range $[0, 1]$ and then reported as percentages. Higher values indicates higher performance. **(ii) Weighted Distance to Optimum (DTO$_w$)**: To evaluate the cost-performance trade-off, we adopt a metric from the fairness literature proposed by Han et al. (2022), which computes the Euclidean distance from a routing architecture at a pre-specified cost penalty to a *utopian point* (i.e., an oracle router). Following the adaptation by Leteno et al. (2025), we use performance on a 0-100 scale and define the inverse cost as $\mathbf{C} = (1 - C) \times 100$ to compute DTO. Since our primary objective is to maintain performance while minimizing cost (we prefer a better-performing model at higher cost over a worse-performing model at lower cost), we propose a weighted version of DTO:

$$\text{DTO}_w(M) = \sqrt{w_c(\mathbf{C}_u - \mathbf{C}_R)^2 + w_p(P_u - P_R)^2} \tag{13}$$

where w_c and w_p are the weights for cost and performance respectively, C_u and P_u represent the utopian (oracle) cost and performance, and C_R and P_R denote the cost and performance of a router. We set $w_c = 0.25$ and $w_p = 0.75$ to emphasize performance over cost in our evaluation. Lower DTO$_w$ indicates higher proximity with the oracle router. **(iii) Relative Cost Difference** (RelDiff): The relative cost difference measures how a router achieves a target performance at lower cost compared to simply routing to the best model of a training dataset. It is calculated as:

$$\text{RelDiff} = \frac{C_{\text{router}} - C_{\text{best}}}{C_{\text{best}}}, \tag{14}$$

where C_{router} is the cost required by the router to achieve a target performance level (e.g., 95% or 100% of the best model's performance), and C_{best} is the cost of always using the best-performing model. Negative values indicate the router is more cost-efficient, while positive values indicate it is less efficient. If the router cannot reach the target performance, $\text{RelDiff} = +\infty$. **(iv) MSE**: This determines whether the strategy adequately estimates the true performance of the candidates. A lower value indicates a better ability to estimate true performance. **(v) NDCG@all**: An optimal routing would always choose the best candidate if the estimated ranking is the same as the true one. A value closer to 1 indicates a better matching to the true ranking.

4.2 Routing Strategies

The different routing strategies evaluated are: **(a) Random Router**: A baseline that selects a routing candidate uniformly at random from the available models. **(b) Oracle Router**: An upper-bound baseline that assumes access to the ground-truth performance of all models. At each step, it selects the model $M^* \in \mathcal{M}$ that achieves the highest true performance while incurring the lowest possible cost, thus representing the ideal cost–quality trade-off. **(c) Best, most expensive and cheapest LLM on the dataset** These baselines correspond to always selecting a single fixed LLM from the dataset: *Best Training LLM*: The model achieving the highest average performance on the training dataset regardless of cost. *Most Expensive LLM*: The model with the highest inference cost. *Cheapest LLM*: The model with the lowest inference cost. **(d) Low-Resource Generalisable Routing Strategies**: CONTEXTUALROUTER and *Universal Model Router* (*UMR*) (Jitkrittum et al., 2026). **(e) K-NN Regression**: The estimation of the performance of each candidate is made by averaging the performance of that candidate across all k neighbours without weighting by distance. Additional implementation details are provided in Appendix A.

4.3 Experiments

4.3.1 Main Experiment

To evaluate the ability of different architectures to perform routing with a fixed candidates pool, we divide each benchmark into training, validation, and test splits. The different architectures are tested on the test splits using all available routing candidates. All queries are encoded using *snowflake-arctic-embed-m-v2.0* as embedding function ϕ (Yu et al., 2025), one of the top performing embedding

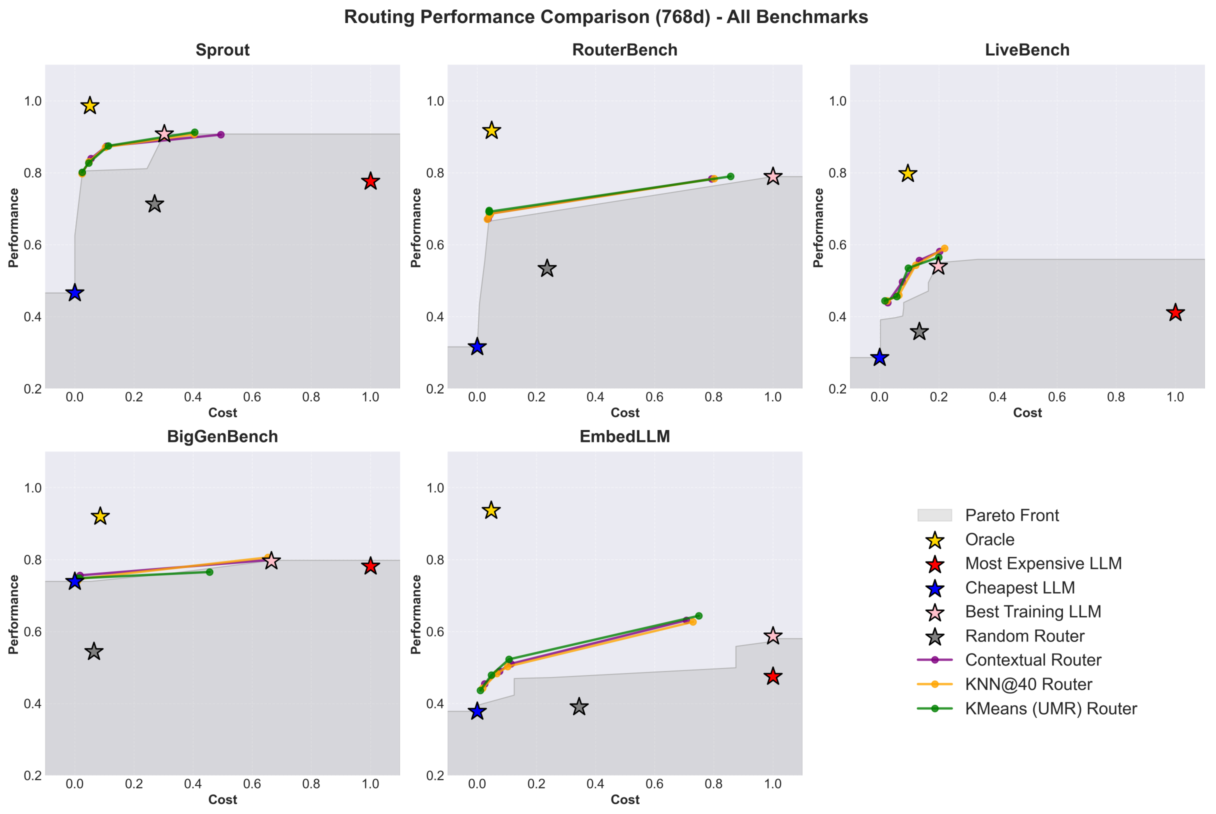


Figure 2: Deferral curves for different routing architectures across varying cost penalties ($\lambda \in \{0, 0.5, 1, 2\}$). The x-axis and y-axis represent the mean performance and mean cost on the benchmark, respectively. Stars denote baseline comparisons. The gray area represents the Pareto front constructed from the mean performance and cost of all individual LLM.

models under 500M parameters on the MTEB retrieval task (Muennighoff et al., 2023). The train split is used to construct the historical performance dataset. We evaluate the routing architectures with different cost penalties: $\lambda \in \{0, 0.5, 1, 2\}$. Additionally, we construct a Pareto front from the mean performance and cost of all individual candidates on the test dataset to evaluate the cost-performance efficiency of the routing strategies against what is achievable using the most optimal single candidates.

First, our results show that on almost all benchmarks, using a routing architecture is a cost-efficient strategy to achieve the same performance as the most performant candidate in the training dataset, or 95% of its performance (Figure 2 and Table 1. See Appendix D for the detailed DTO$_w$ and peak performance). For example, in the *EmbedLLM* benchmark, it lowered the cost by half while achieving the same performance. In most benchmarks, the performance-cost curves of the different routing strategies lie above the Pareto front constructed from individual LLMs cost and performance, indicating that these routing strategies are more efficient than using any single routing candidate alone (Figure 2). However, in most settings, neither our architecture nor *UMR* outperformed significantly the K-NN@40 algorithm, whether in terms of DTO$_w$, peak performance, MSE, or NDCG (Tables 5 and 2). In fact, recent work showed that K-NN was a strong baseline, frequently outperforming more complex learned strategies (Li, 2025; Yuan et al., 2025). Similarly to Yuan et al. (2025), we did not find that *UMR* (Jitkrittum et al., 2026) outperforms the K-NN. Our results might differ because they used another embedding model, *Gecko* (Lee et al., 2024; Jitkrittum et al., 2026).

4.3.2 Impact of the Nearest Neighbours Size

As the number of nearest neighbours directly determines the amount of information available to estimate candidate performance, tuning k has a significant impact on the model's results. A high value of k may introduce noise by including queries that are too dissimilar to reflect the true local performance of the LLM candidate. A low value of k may result

Benchmark	C-ROUTER		*K-NN*		*UMR*	
	@95%	@100%	@95%	@100%	@95%	@100%
Sprout	**-70.5**	$+\infty$	-69.9	$+\infty$	-68.3	20.5
RouterBench	-46.6	$+\infty$	-46.2	$+\infty$	**-49.0**	**-14.9**
LiveBench	-53.5	-40.4	-49.1	-39.8	**-56.7**	**-43.0**
BigGenBench	**-96.6**	-7.2	-81.7	**-17.6**	-66.6	$+\infty$
EmbedLLM	-65.2	-50.8	-61.9	-47.1	**-70.6**	**-55.0**

Table 1: Relative cost difference (RelDiff) to achieve 95% (@95%) and 100% (@100%) of the best LLM performance on the training dataset ($\lambda = 0$, dimension=768). Positive values indicate the router requires higher cost to reach the target performance; negative values indicate the router achieves the same performance at lower cost. If the router cannot reach the target performance, RelDiff $= +\infty$. C-ROUTER: CONTEXTUALROUTER

Benchmark	C-ROUTER		*K-NN*		*UMR*	
	MSE	NDCG	MSE	NDCG	MSE	NDCG
Sprout	**0.126**	94.0	0.128	94.0	0.133	**94.1**
RouterBench	**0.157**	88.2	**0.157**	88.2	**0.157**	**88.5**
LiveBench	**0.167**	77.5	0.172	**77.6**	0.180	77.1
BigGenBench	**0.047**	**96.1**	0.049	**96.1**	0.054	95.3
EmbedLLM	**0.187**	76.3	0.188	75.8	**0.187**	**76.5**

Table 2: Performance of routing architectures on different benchmarks ($\lambda = 0$, dimension=768). Lower MSE indicates more accurate performance prediction. NDCG is expressed as a percentage, with higher values indicating better matching with true ranking order. *ContRout:* CONTEXTUALROUTER

in insufficient information to correctly estimate the performance of the candidate. Thus, we train and evaluate routing performance (mean performance and mean cost) across different values of k, while keeping the cost penalty at $\lambda = 0$ at inference. The different k tested are $k \in \{10, 20, 40, 80\}$.
Increasing the number of neighbours beyond k = 40 does not result in a significant improvement in either DTO$_w$ or peak performance. This may be because relevant past performance is easily retrieved by the embedding model. *MMLU* items, for instance, are highly semantically similar to one another, which makes it relatively easy to retrieve relevant information. The difficulty of the retrieval task may be an important confounding factor in our results. We also observe a trade-off between cost efficiency and performance. Smaller values of k tend to have better cost–performance ratios (lower DTO$_w$), whereas larger k has higher peak performance. As more examples are retrieved, larger and more expensive LLMs with more generalistic ability may drive the performance retrieved, thereby masking the localized strengths of smaller models. Overall, k is a critical hyperparameter: tuning it can improve either DTO$_w$ or peak performance relative to the default setting of $k = 40$ or *UMR*.

4.3.3 Impact of the Embedding Model Size

The strategies rely on embeddings adequately representing query semantics. Thus, we evaluate models of different sizes to assess whether this impacts the ability to estimate performance more accurately and, indirectly, routing performance. We evaluate three types of embedding model of different number of dimensions and architectures : (i) *potion-multilingual-128M*, a static embedding model distilled from *BAAI/bge-m3* sentence transformer (256d)[1]; (ii) *snowflake-arctic-embed-m-v2.0*, a sentence transformer of 305M parameters (768d) (Yu et al., 2025); and (iii) *text-embedding-3-large*, a proprietary embeddding model from *OpenAI* (3072d)[2].
Using different sizes of embedding models does not result in significant differences. In fact, even the static embedding model achieves almost the same performance as the most efficient one, Snowflake's embedding model (Yu et al., 2025). The figures are available in the appendix D and Figure 2.

[1]https://huggingface.co/blog/Pringled/model2vec
[2]https://openai.com/index/new-embedding-models-and-api-updates/

4.3.4 Performance and Generalization in Low-Data Settings

In low-resource environments, there may be limited annotated history performance data available for routing. To ensure an unbiased assessment, we used *FusionBench* (Feng et al., 2025b), a dataset that has not been used in CONTEXTUALROUTER training. We partition the dataset into 85% training and 15% test splits. CONTEXTUALROUTER retrieves from available data without retraining, while *UMR* reconstructs clusters for each data size (Jitkrittum et al., 2026). We vary available training data by sampling 1%, 2.5%, 5%, 7.5% and 10% of the training split and evaluate mean performance and cost under cost penalty setting of $\lambda = 0$ on the test dataset. We tested two scenarios: 1) training and evaluating on the same 50% candidate subset, and 2) training on 50% then adding remaining candidates at inference. For *UMR* (Jitkrittum et al., 2026), in the second scenario, we optimise cluster selection on half the candidates, then incorporate new candidates into existing clusters at inference.
The results demonstrate that CONTEXTUALROUTER and K-NN@40 remain unaffected by the limited amount of available historical data. Even with only 1% (n=53) samples available, these architectures achieve the same performance as with 10% (n=535). However, our architecture does not outperform the K-NN@40. *UMR* (Jitkrittum et al., 2026) underperformance with 1% available data may be caused by the construction of very small and noisy clusters, which result in a performance representation of a candidate that is too different from its true one (Appendix D). The results are similar whether tested on the sample of candidates used during training or on additional ones introduced afterward. All these architectures are able to generalise easily to new candidates; however, K-NN@40 might be sufficient to achieve adequate performance even with a small amount of historical performance data. A downside of these unsupervised approaches is the need to maintain a query corpus with LLM performance records at inference, however, only very small corpora are needed, making these methods realistically applicable in practice.

4.3.5 Real-Case Simulation

Each dataset uses highly similar question formats across samples (e.g., multiple-choice questions in the *MMLU* dataset). This lack of diversity makes the retrieval task relatively easy. To simulate real-world user queries (which are often concise, informally phrased, and inconsistently cased), we sampled 3000 examples from each of the three benchmarks with the highest number of samples: *RouterBench*, *EmbedLLM*, and *Sprout*. Each sample was paraphrased using *gpt-4o-mini* to mimic realistic query variations. The paraphrasing prompt with examples are available in Appendix C. The figure in Appendix D shows how paraphrasing induces significant overlap between the datasets composing each benchmark. For each benchmark, we selected 1000 samples for training (*UMR*) or retrieval (K-NN@40 and CONTEXTUALROUTER), and used the remaining 2000 samples for testing. For CONTEXTUALROUTER, we use the version trained in section 4.3.1. To evaluate the impact of retrieval degradation, we report the difference in peak performance (Δ_{Peak}) and DTO$_w$ (Δ_{DTO_w}) between the routers applied on the original and paraphrased samples.
We observe limited performance degradation when simulating real-world queries to degrade retrieval with $-0.3 \leq \Delta_{\text{Peak}} \leq 2.0$ and $-3.0 \leq \Delta_{\text{DTO}_w} \leq 1.1$. While *EmbedLLM* shows improved peak performance, this comes with increased DTO$_w$ for CONTEXTUALROUTER and *UMR*, but not K-NN@40. Overall, these findings indicate that these low-resource unsupervised routing strategies are robust to the distributional shift introduced by realistic query simulations. This also suggests that the influence of benchmark semantic similarity on routing performance may be less significant than hypothesised in previous sections. See Appendix C for tables.

5 Conclusion

Our results demonstrate that unsupervised routing algorithms provide a cost-effective and generalisable alternative to traditional monolithic single LLM architectures. They decrease financial and computational requirements while maintaining comparable performance. Consistent with previous work (Li, 2025), even vanilla K-NN is as effective as learned retrieval-based routers on benchmarks. These findings challenge the necessity of LLM-based routing solutions (Zhang et al., 2025) for generalisable routing. Furthermore, we showed that these strategies require only a small corpus of queries to be effective. This opens up new deployment opportunities in environments with limited computational resources.

Limitations

As demonstrated with *MMLU* (Gema et al., 2025), many datasets contain flaws including unanswerable questions, multiple valid answers, and incorrect ground truth. For instance, in *SPROUT*'s *Teknium* sub-dataset, candidates are penalized for valid responses that differ from ground truth despite correctly following open-ended instructions (see Appendix B for various examples). Incorrect evaluation affects both training by penalizing correct models and inference by giving contradictory historical performance data. There is a need to evaluate the proportion of the difference between tested routing algorithms and the oracle router that does not stem from inherently unpredictable candidate performance. Yuan et al. (2025) analysed several benchmarks used in this work, i.e. *EmbedLLM* and *RouterBench*, and identified a number of issues. One issue is *LLM dominance*, whereby a single candidate model may outperform most others. We observe this in the *Sprout* benchmark: *gpt-o3-mini* is sufficiently general-purpose and cheap that it often becomes the best option overall, making routing unnecessary. In this case, the focus shifts from selecting the most suitable LLM for each task to identifying the best overall model, which is closer to a multi-armed bandit setting. Another issue identified is *LLM ability redundancy*, where several candidate LLMs exhibit similar capabilities at similar costs, adding noise to the routing problem (Yuan et al., 2025). This challenges the assumption that very large pools of LLMs are necessary, as serving hundreds of models may be inefficient in real-world deployments. While Yuan et al. (2025) show that K-NN achieves the strongest performance on these benchmarks cleaned using strategies designed to remove these issues, they may have affected the training of learned routing strategies such as CONTEXTUALROUTER. In a real-world deployment, routers will be implemented in an online sequential or batch setting. Consequently, a retrieval-based router requires an incremental corpus of history data at inference time. Future work should therefore study the most efficient methods for constructing and maintaining such a corpus in this setting. Additionally, it remains an open question whether retrieval-based routers suffer from cold-start issues when the corpus size is small (inferior to k).

References

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A complete survey on LLM-based AI chatbots. *Preprint*, arXiv:2406.16937.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. 2025. A unified approach to routing and cascading for LLMs. *Preprint*, arXiv:2410.10347.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The 12th International Conference on Learning Representations*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025a. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations*.

Tao Feng, Haozhen Zhang, Zijie Lei, Pengrui Han, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Jiaxuan You. 2025b. Fusing LLM capabilities with routing data. *Preprint*, arXiv:2507.10540.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. 2025. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5069–5096, Albuquerque, New Mexico. Association for Computational Linguistics.

Xudong Han, Timothy Baldwin, and Trevor Cohn. 2022. Balancing out bias: Achieving fairness through balanced training. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11335–11350, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024a. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024b. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*.

Swayambhoo Jain, Ravi Raju, Bo Li, Zoltan Csaki, Jonathan Li, Kaizhao Liang, Guoyao Feng, Urmish Thakkar, Anand Sampat, Raghu Prabhakar, and Sumati Jairath. 2024. Composition of experts: A modular compound AI system leveraging large language models. *Preprint*, arXiv:2412.01868.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 7036–7050, Mexico City, Mexico.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. 2026. Universal model routing for efficient LLM inference. In *The Fourteenth International Conference on Learning Representations*.

Seungone Kim, Juyoung Suk, Ji Yong Cho, Shayne Longpre, Chaeeun Kim, Dongkeun Yoon, Guijin Son, Yejin Cho, Sheikh Shafayat, Jinheon Baek, Sue Hyun Park, Hyeonbin Hwang, Jinkyung Jo, Hyowon Cho, Haebin Shin, Seongyun Lee, Hanseok Oh, Noah Lee, Namgyu Ho, and 13 others. 2025. The BiGGen bench: A principled benchmark for fine-grained evaluation of language models with language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5877–5919.

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. 2024. Gecko: Versatile text embeddings distilled from large language models. *Preprint*, arXiv:2403.20327.

Thibaud Leteno, Michael Perrot, Charlotte Laclau, Antoine Gourru, and Christophe Gravier. 2025. Fair text classification via transferable representations. *Preprint*, arXiv:2503.07691.

Yang Li. 2025. Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers. *Preprint*, arXiv:2505.12601.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. 2024. OptLLM: Optimal assignment of queries to large language models. *Preprint*, arXiv:2405.15130.

Mohammadhossein Malekpour, Nour Shaheen, Foutse Khomh, and Amine Mhedhbi. 2024. Towards optimizing SQL generation via LLM routing. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

Niklas Muennighoff, Nouamane Tazi, Loic Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, Dubrovnik, Croatia. Association for Computational Linguistics.

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

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2024. Large language model routing with benchmark datasets. *Preprint*, arXiv:2309.15789.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. 2025. CARROT: A cost aware rate optimal router. In *ICLR 2025 Workshop on Foundation Models in the Wild*.

Kv Aditya Srivatsa, Kaushal Maurya, and Ekaterina Kochmar. 2024. Harnessing the power of multiple minds: Lessons learned from LLM routing. In *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 124–134, Mexico City, Mexico. Association for Computational Linguistics.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. 2024. TensorOpera router: A multi-model router for efficient LLM inference. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 452–462, Miami, Florida, US. Association for Computational Linguistics.

Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. 2024. Learning to defer to a population: A meta-learning approach. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 3475–3483. PMLR.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D. Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of LLMs. *Preprint*, arXiv:2501.06322.

Joaquin Vanschoren. 2018. Meta-learning: A survey. *Preprint*, arXiv:1810.03548.

Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. 2025. Doing more with less: A survey on routing strategies for resource optimisation in large language model-based systems. *Preprint*, arXiv:2502.00409.

Xuanhui Wang, Cheng Li, Nadav Golbandi, Mike Bendersky, and Marc Najork. 2018. The lambdaloss framework for ranking metric optimization. In *Proceedings of The 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*, pages 1313–1322.

Yuanshuai Wang, Xingjian Zhang, Jinkun Zhao, Siwei Wen, Peilin Feng, Shuhao Liao, Lei Huang, and Wenjun Wu. 2024. Bench-CoE: a framework for collaboration of experts from benchmark. *Preprint*, arXiv:2412.04167.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. Livebench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel F Campos. 2025. Arctic-embed 2.0: Multilingual retrieval without compromise. In *Second Conference on Language Modeling*.

Jiayi Yuan, Yifan Lu, Rixin Liu, Yu-Neng Chuang, Hongyi Liu, Shaochen Zhong, Yang Sui, Guanchu Wang, Jiarong Xing, and Xia Hu. 2025. Who routes the router: Rethinking the evaluation of LLM routing systems. In *NeurIPS 2025 Workshop on Evaluating the Evolving LLM Lifecycle: Benchmarks, Emergent Abilities, and Scaling*.

Haozhen Zhang, Tao Feng, and Jiaxuan You. 2025. Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. 2025. EmbedLLM: Learning compact representations of large language models. In *The Thirteenth International Conference on Learning Representations*.

A Implementations

A.1 Settings

We define LLM cost as the input price per million tokens, using published values where available. For models without published pricing, we estimate cost from parameter count using TogetherAI pricing[3]. Both costs and benchmark performance scores are normalized across all routing candidates.

A.2 ContextualRouter

We trained the model using the *AdamW* optimizer with a learning rate of $1e^{-4}$, weight decay of $1e^{-5}$, and batch size of 64. We applied a learning rate scheduler that reduced the rate by a factor of 0.5 when validation loss plateaued. Since ContextualRouter is agnostic to the number of candidates, we shuffled batches across benchmarks to train on diverse routing contexts simultaneously. We also shuffled neighbours and candidates within each batch to encourage learning generalisable performance patterns rather than candidate-specific behaviors. For the 3072-dimensional embedding model, we added dropout layers to the attention matrix and feedforward network to prevent overfitting, and adjusted the learning rate to $1e^{-5}$ and weight decay to $1e^{-3}$.

A.3 Universal Model Routing (Jitkrittum et al., 2026)

The original approach proposed by Jitkrittum et al. (2026) search the number of clusters which maximises the area under the deferral curve (cost-performance curve) for different cost penalty. This approach leverage certain complexity as multiple runs on possible on large datasets are necessary to find the most adequate k, specifically when the range of k values is large. Instead, we search for a specified cost penalty, the number of clusters which maximise the mean performance and minimise the mean cost on a validation dataset. We did not implement their soft clustering approach as it is per nature not generalisable to new candidates as the proper clustering is learnt from specific candidates performances.

[3]https://www.together.ai/pricing (accessed 06/05/2025)

B Errors in Datasets - Example of Sprout's OpenHermes\Teknium subdataset

We examine several errors found in one of the routing benchmark:

B.1 Incorrect Task Evaluation

Prompt: Summarize a recent science breakthrough in any field, explaining its significance and potential impact on society.
Annotated Ground Truth: [*Summary about AlphaFold*]

In this case, there is no single correct answer. The task requires demonstrating the ability to summarize a recent scientific breakthrough. However, in the dataset, all candidates generated adequate summaries (e.g., on nuclear fusion or CRISPR) but were scored as erroneous because their summaries did not match the specific topic of the ground truth.

B.2 Incorrect Ground Truth

Prompt: Which ancient civilization is credited with inventing the concept of zero in mathematics?
Annotated Ground Truth: The Mayans

The correct answer should be "Ancient Indians." All candidate responses provided this correct answer, but the LLM-as-a-Judge assigned a score of 0 to every candidate.

B.3 Ambiguous Prompts

Prompt: Who was the British Prime Minister at the start of World War II, famously known for his speech "We shall fight on the beaches" ?
Annotated Ground Truth: Neville Chamberlain

This example illustrates a case where the question has no true answer. Neville Chamberlain was indeed the British Prime Minister at the start of World War II, but the famous "We shall fight on the beaches" speech was delivered by Winston Churchill.

C Simulating real-world queries

To simulate real-world user queries, we paraphrased benchmark questions into informal queries that mimic how users interact with chatbots. This section presents the paraphrasing prompt and examples of the transformation.

C.1 Paraphrasing Prompt

The following prompt was used to paraphrase each sample:

You will be given benchmark questions. Your task is to transform these questions into realistic queries that mimic how real users interact with modern chatbots while preserving all the original information.

Guidelines:
1. **Preserve All Information**: Every fact, concept, and detail from the original question must remain in your output.
2. **Maximum 10 Words**: Your output must be 20 words or fewer. If possible, as concise as possible. Must be equivalent or smaller than original question length
3. **Mimic Real Chatbot Users**: Transform questions to reflect authentic user behavior:
- Degrade the syntax, do not capitalize - Vague references ("that thing," "the stuff") - Context-less requests assuming the AI understands - Overly brief or overly verbose extremes - Multiple questions or tangents in one query - Assumptions or incomplete thoughts
4. **Modern Chatbot Patterns**: Include realistic user behaviors like: - Casual curiosity without formality - Mobile-style typing (short, fragmented)
5. **Do NOT include answer choices** in the output—only transform the question itself and do NOT mention them: do not output a, b, c or d?

Only return the reformulated questions.

C.2 Paraphrasing Example (Sprout/MMLU-Pro)

You are an knowledge expert, you are supposed to answer the multi-choice question to derive your final answer as ‘The answer is ...‘
Q: Why does milk spoil when kept in a refrigerator? Options are:
(A): Milk spoils due to high temperatures
(B): Milk spoils exclusively due to the separation of fat content at lower temperatures
(C): Milk spoils because of thermophilic bacteria
(D): Psychrophilic bacteria cause milk to spoil in cool temperatures.
(E): Milk spoils as a result of exposure to light inside the refrigerator
(F): Spoilage occurs due to an increase in pH levels over time, regardless of temperature
(G): Milk spoils due to chemical reactions
(H): Milk spoils due to oxygenation through the container's permeability
(I): Enzymatic activity from feed consumed by cows causes milk to spoil in the fridge
(J): Milk spoils because of the absorption of refrigerator odors

It turns into: **why does milk still spoil even in the fridge, what causes it exactly?**

C.3 Results of the experiment

Benchmark	C-ROUTER	KNN	UMR
Sprout	+0.0	-0.3	-0.1
RouterBench	+0.2	+0.0	+0.1
EmbedLLM	+2.0	+1.3	+1.0

Table 3: $\Delta_{\textbf{Peak}}$ after paraphrasing (Parameters: $\lambda = 0$, dimension=768). Positive values indicate performance improvement. C-ROUTER: CONTEXTUALROUTER

Benchmark	C-ROUTER	KNN	UMR
Sprout	-0.6	-0.7	-1.5
RouterBench	+0.4	+1.1	-3.0
EmbedLLM	+0.4	-0.2	+1.1

Table 4: $\Delta_{\textbf{DTO}_w}$ after paraphrasing (Parameters: $\lambda = 0$, dimension=768). Negative values indicate improvement. C-ROUTER: CONTEXTUALROUTER

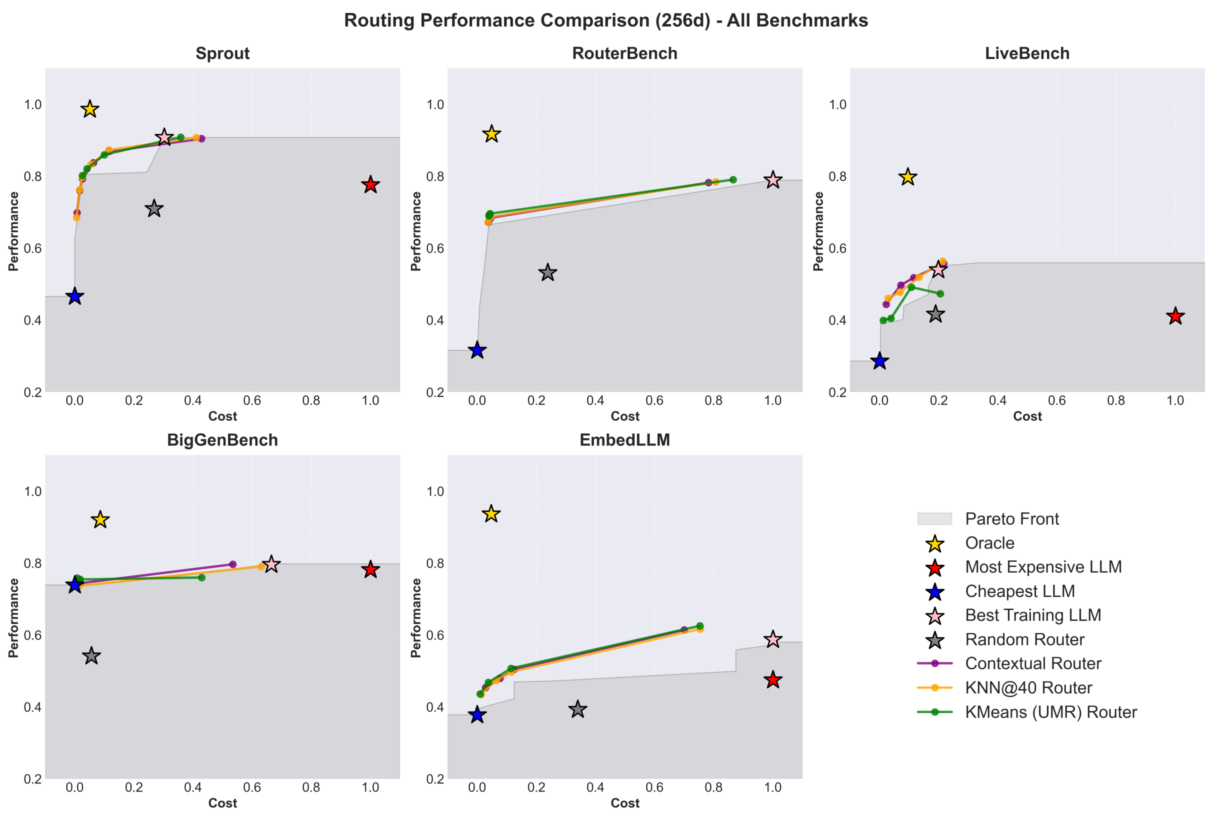
D Additional Figures and Tables

Strategy	Sprout		RouterBench		LiveBench		BigGenBench		EmbedLLM	
	DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak
Oracle	0.0	98.6	0.0	91.7	0.0	79.7	0.0	92.0	0.0	93.7
Most Exp. LLM	50.8	77.6	48.9	78.9	56.3	41.0	47.3	78.1	62.2	47.5
Cheapest LLM	45.1	46.6	52.1	31.6	44.5	28.6	16.2	73.9	48.5	37.8
Best Tr. LLM	**14.3**	90.7	48.9	78.9	22.8	54.0	30.9	79.6	56.5	58.7
Random Router	26.1	71.3	**34.6**	53.3	38.1	35.8	32.6	54.4	49.6	39.0
C-Router	23.2	90.6	39.0	78.3	19.5	58.1	30.6	79.9	**42.4**	63.1
K-NN@40	18.9	90.7	39.4	78.3	**19.0**	**59.0**	**30.1**	**80.6**	43.4	62.7
KMeans (UMR)	18.8	**91.3**	41.9	**79.0**	20.8	56.5	22.9	76.5	43.3	**64.4**

Table 5: Comparison of routing strategies across benchmarks. **Bold** indicates best performance after oracle routing, underline indicates second best. All three generalisable routing architectures show strong efficiency-performance trade-offs across benchmarks ($\lambda = 0$, dimension=768). *Peak: Maximum performance achieved. DTO$_w$: Weighted Distance to Optimum*; C-Router: ContextualRouter; *Best Tr. LLM: Best Training LLM*; *Most Exp. LLM: Most Expensive LLM*

Strategy	k	Sprout		RouterBench		LiveBench		BigGenBench		EmbedLLM	
		DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak	DTO$_w$	Peak
C-Router	*10*	22.5	88.8	**32.0**	74.4	24.5	52.0	**25.0**	75.9	**40.9**	59.0
	20	**22.2**	90.2	35.6	76.9	21.0	56.2	28.3	77.3	41.5	61.1
	40	23.2	**90.6**	39.0	78.3	**19.5**	**58.1**	30.6	79.9	42.4	63.1
	80	25.1	90.5	40.3	**79.0**	**19.5**	57.8	29.5	**80.0**	42.7	**63.8**
K-NN	*10*	23.2	88.8	**32.2**	73.9	22.8	53.9	**24.5**	76.9	**41.2**	58.8
	20	20.3	90.2	35.9	76.9	**18.5**	**59.3**	29.9	79.1	41.9	61.3
	40	18.9	**90.7**	39.4	78.3	19.0	59.0	30.1	**80.6**	43.4	62.7
	80	**18.4**	90.9	41.1	**79.3**	20.7	56.3	32.2	79.7	44.4	**63.2**

Table 6: Impact of neighborhood size k on neighbours-based routing strategies performance across benchmarks. **Bold** indicates best k performance on a specific routing strategy, underline indicates second best. Results use $\lambda = 0$, dimension=768. *Peak: Maximum performance achieved at each k value; DTO$_w$: Weighted Distance to Optimum;* C-Router*:* ContextualRouter



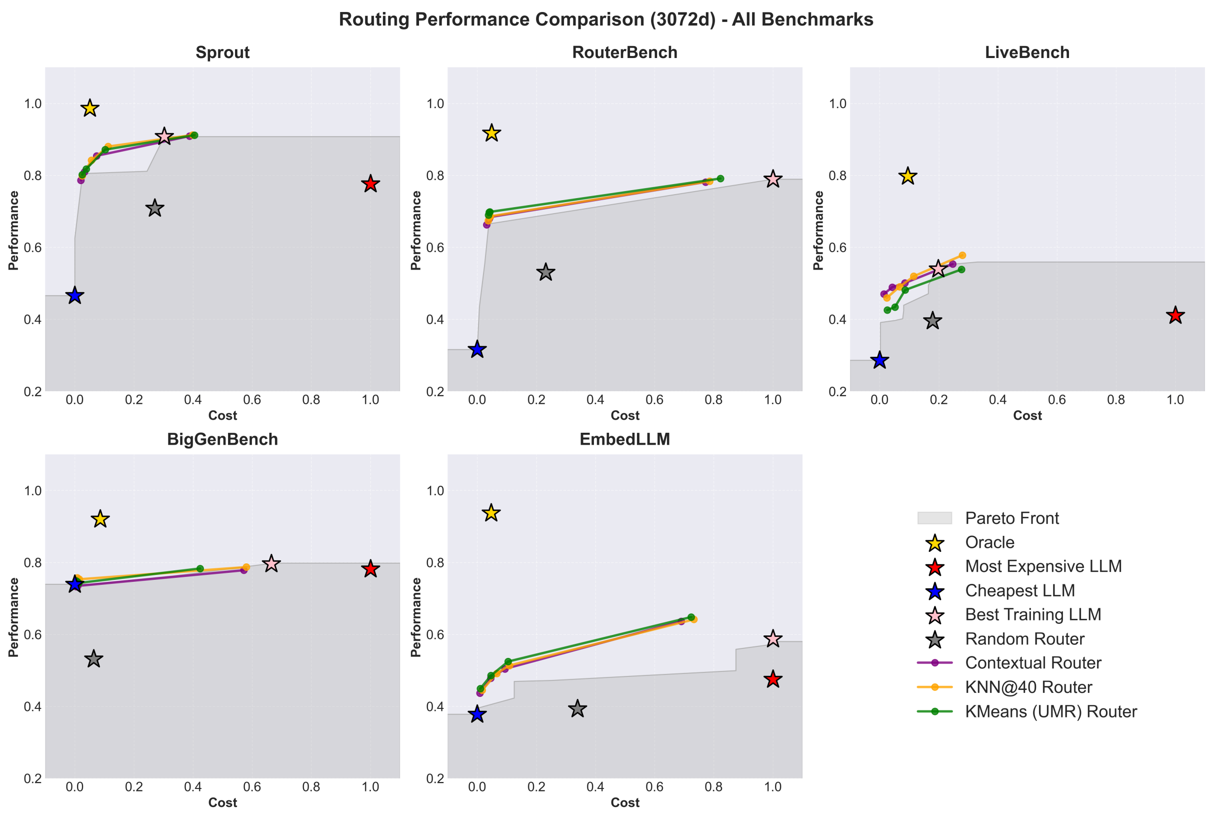


Figure 3: Deferral curves of the different routing architectures for different cost penalty ($\lambda \in \{0, 0.5, 1, 2\}$). It represents the experiment with the 256 and 3072-dimensions embedding models. The different stars represent the baseline comparisons. The gray area represents the Pareto front constructed from the mean performance of all individual routing candidates.

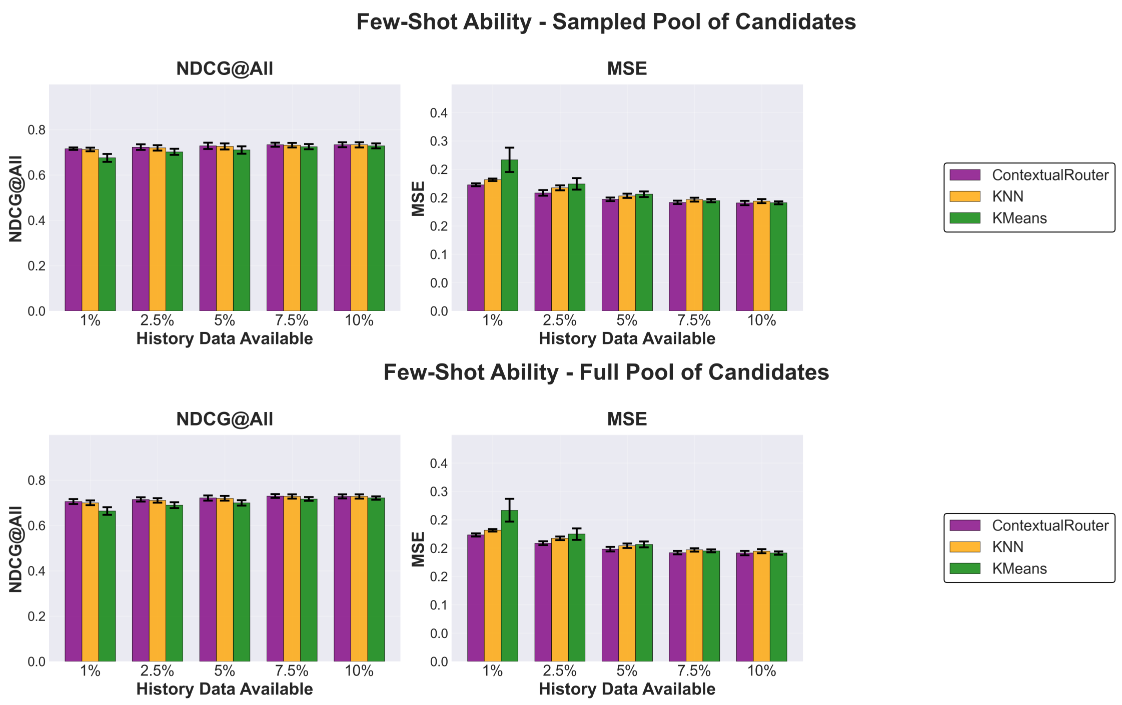


Figure 4: Results of the ability to accurately estimate performance of candidates on low amount of data and to generalise to new candidates in this context. The results are aggregated from 10 runs. *n: size of history data available. KMeans: UMR (Jitkrittum et al., 2026)*

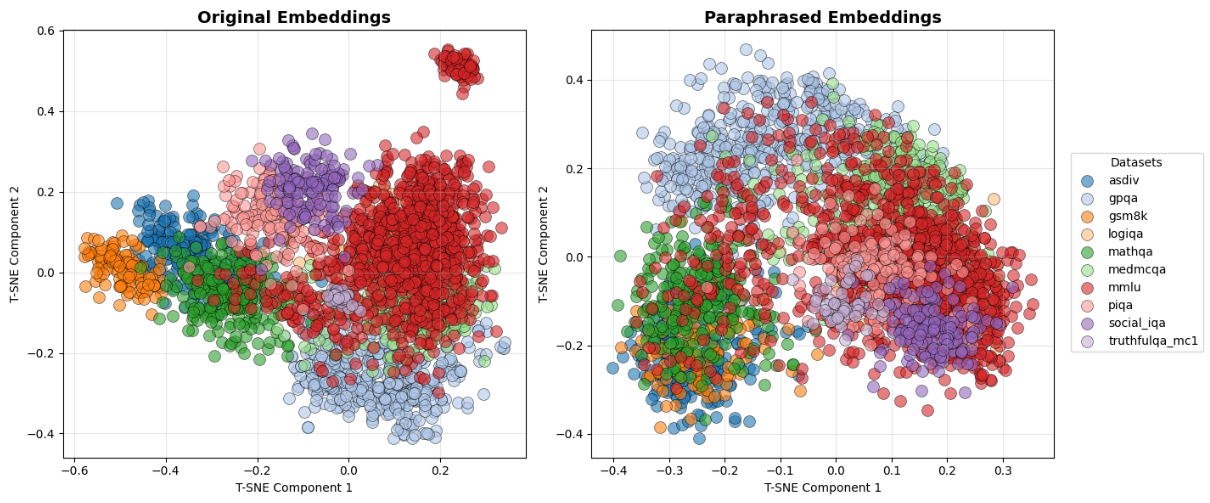


Figure 5: Influence of paraphrasing on the distribution of *EmbedLLM* datasets samples across semantic space. Original and reformulated queries are embedded and visualized in 2D using t-SNE. Before paraphrasing, each dataset is visually distinct; after paraphrasing, substantial overlap occurs across datasets.