ProxRouter: Proximity-Weighted LLM Query Routing for Improved Robustness to Outliers

Shivam Patel[1†] **Neharika Jali**[1] **Ankur Mallick**[2] **Gauri Joshi**[1]
[1]Carnegie Mellon University, [2]Microsoft

Abstract

Large language model (LLM) query routers are critical to modern AI platforms as they seek to improve efficiency by assigning inference queries to accurate, yet low-cost models. Parametric routers typically use trained neural networks for LLM selection but suffer from retraining and maintenance overheads. Nonparametric routers are training-free, instead estimating LLM accuracy and cost via similarity between encodings of the input query and training set queries. However, like their parametric counterparts, nonparametric routers struggle to generalize to outlier queries, an issue exacerbated by limited diversity in training sets which are costly to expand and difficult to keep current with ever-evolving use cases. We propose ProxRouter, which applies an exponentially tilted aggregation mechanism to balance bias and variance in nonparametric routers, improving their robustness to outliers. Experiments show ProxRouter enhances outlier routing while preserving inlier performance with minimal overhead.

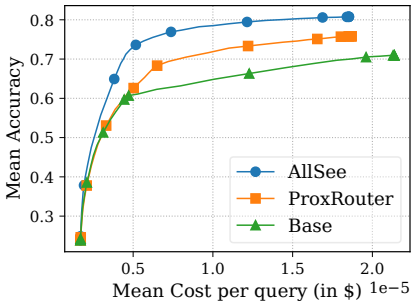


Figure 1: The *Base* router (a nearest neighbors router for this plot), which is only trained on inlier queries, fails to generalize to outlier tasks at test time, showing upto 15% average accuracy drop for a given cost, relative to the *AllSee* router trained on both inliers and outliers. Our proposed *ProxRouter*, although trained only on inlier queries, improves robustness to outliers and achieves a better accuracy-cost trade-off (experimental details in Section 4).

1 INTRODUCTION

Generative AI, particularly large language models (LLMs) Brown et al. (2020); Vaswani et al. (2017), has achieved remarkable breakthroughs, surpassing human-level accuracy on a wide range of tasks OpenAI (2024). Yet, state-of-the-art models contain billions or even trillions of parameters Kaplan et al. (2020), making inference excessively costly. As LLMs are integrated into an expanding set of applications, the landscape has rapidly diversified, with models of varying sizes, costs, and levels of specialization. The open-source ecosystem alone hosts hundreds of thousands of pretrained and fine-tuned models[1], spanning a broad spectrum of capabilities Hu et al. (2022). Using a frontier-scale model for every query is often unnecessary, particularly when smaller, cheaper models can deliver equally accurate responses to simpler inputs. In fact, domain-specific lightweight models frequently outperform general-purpose LLMs on specialized downstream tasks while operating at a fraction of the cost Touvron et al. (2023). This combination of model abundance and the high expense of large-scale inference has driven growing interest in *efficient query routing*, which seeks to dynamically select the most appropriate model to deliver accurate responses at minimal cost.

Query routing seeks to identify the most suitable model from a large pool of LLMs, minimizing inference costs while maintaining response quality. A central challenge is estimating the accuracy and cost of responses to unseen queries. Most methods begin with fixed-

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

[1]As of this writing, over 282,000 models are available on huggingface.co/models.

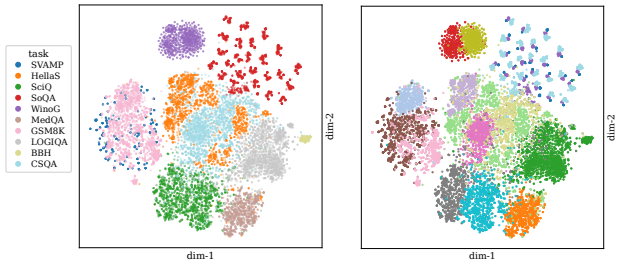


Figure 2: High-dimensional query encodings downprojected using t-SNE van der Maaten and Hinton (2008). *Left*: colored by task. *Right*: colored by cluster assignment through KMeans clustering ($K = 16$). Queries from same task occupy compact, localized neighborhoods in encoding space, allowing clustering to recover semantically coherent regions aligned with query types.

dimensional query encodings generated by deterministic sentence encoder models Cer et al. (2018); Reimers and Gurevych (2019), which form the backbone of both parametric and nonparametric routers. Parametric routers train small neural networks (e.g., multi-layer perceptrons) to predict correctness or cost Hu et al. (2024); Ding et al. (2024, 2025); Zhuang et al. (2024); Sakota et al. (2024); Huang et al. (2025), or leverage ranking and reward models Ong et al. (2025); Zhang et al. (2024); Chen et al. (2025). In contrast, nonparametric routers are training-free and operate directly in the query encoding space. Two canonical forms dominate: clustering-based approaches (e.g., KMeans) Zhang et al. (2025b,a); Jitkrittum et al. (2025b); Hu et al. (2024) and nearest-neighbor methods (e.g., kNN) Li (2025b); Zhuang et al. (2024); Jitkrittum et al. (2025a); Stripelis et al. (2024). Despite their simplicity, nonparametric routers often rival or surpass parametric ones, likely due to the expressive power of encoder-based query embeddings (Figure 2), which make the mapping from queries to model abilities straightforward. Moreover, clustering and nearest-neighbor methods naturally accommodate new queries and models without retraining, a key advantage over parametric designs. *For these reasons, in this work, we focus on the nonparametric routing paradigm.*

The success of query routers depends critically on the quality of their accuracy and cost estimates for incoming queries. High-quality estimates require large and diverse router training datasets in which queries are evaluated on all, or a substantial subset of models in the pool. Running inference on all models for each query is expensive, and thus, routers are typically trained only for a limited set of downstream tasks. As new applications emerge, entirely novel query types (e.g., new programming languages) can lead to poor routing performance (see Figure 1). Even minor changes in phrasing, for instance, appending Chain-of-Thought instructions Wei et al. (2023), can shift both accuracy and cost, further complicating generalization. Retraining the router on these outlier queries will require evaluating those queries on all models in the pool, which can be prohibitively expensive. While some approaches replace encoder-based routers with LLMs (e.g., GPT-4o-mini) to make routing decisions directly, the resulting increase in routing cost is impractical at scale. Thus, although accurate predictions of model correctness across all queries remain the objective, the central challenge lies in achieving robust generalization to outlier and unseen queries. Figure 1 illustrates how poor generalization to a diverse set of queries leads to performance drops, underscoring the need for better robustness.

In this work, we propose ProxRouter, a method that enhances the robustness of clustering- and nearest-neighbor–based nonparametric routers to outlier queries by improving the model accuracy and cost estimates for all queries. The main contributions of our paper are summarized below:

- In Section 2, we develop a unified framework for nonparametric routers, of which clustering and nearest-neighbor routers are special cases. The framework formulates accuracy and cost estimates as a weighted average of estimates provided by reference clusters or neighbors. This enables standardized design and analysis of more general proximity-based routers.
- In Section 3, we introduce ProxRouter, which generalizes clustering and nearest-neighbor routers into proximity-weighted versions, where clusters or neighbors closer to a test query receive higher weights (as illustrated in Figure 3). Aggregation weights are set according to a minimum-variance prior and exponentially tilted to reduce bias. By using soft aggregation over training data, ProxRouter yields more accurate model performance and cost estimates, particularly for outlier queries. ProxRouter does not require any outlier detection and its associated overhead, since it applies the soft aggregation to all queries, both inliers and outliers.
- In Section 4, we extensively evaluate ProxRouter on a diverse set of 14 LLMs and queries from 10 publicly available datasets with numerous outlier-task variations. Results demonstrate that ProxRouter significantly improves robustness to outlier queries while preserving inlier performance. It increases the area under the accuracy–cost curve (AUC), bringing it closer to that of an AllSee router trained on both inlier and outlier queries.

Finally, in Section 5, we summarize our findings and

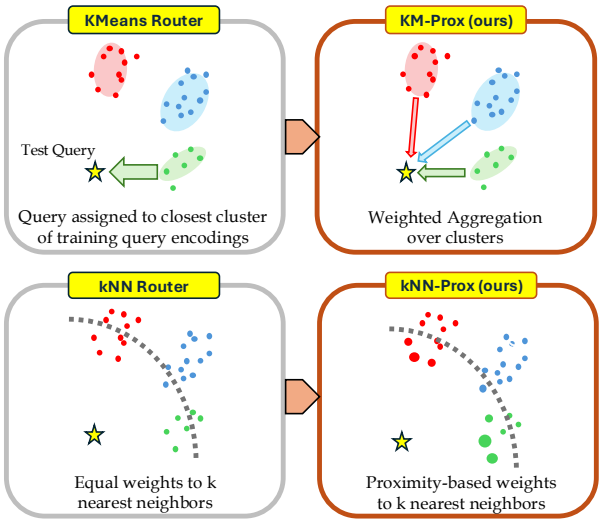


Figure 3: Comparison of our proximity-weighted KMeans clustering and kNN routers. While KMeans assigns a test query encoding to the closest cluster of training query encodings to obtain the accuracy and cost estimates, KM-Prox takes a proximity-weighted combination of the K clusters' for each test query. Similarly, kNN-Prox takes a proximity-weighted combination of the estimates of the k nearest neighbors of the test query.

discuss extensions of the framework to other nonparametric routers. Details of the bias–variance decomposition, more detailed related work, and additional experimental results are deferred to the Appendix.

2 PROBLEM FORMULATION

2.1 Objective of the Query Router

The typical query router's objective is to maximize the accuracy of responses to inference queries subject to cost constraints, or to minimize cost subject to accuracy constraints. We consider a Lagrangian relaxation (Jitkrittum et al. (2025a); Hu et al. (2024); Zhang et al. (2025a); Li (2025b,a)) of the constrained formulation, which results in an unconstrained objective:

$$m^*(\mathbf{x}) \leftarrow \arg\max_{m\in\mathcal{M}}[\ \mathrm{acc}^{(m)}(\mathbf{x}) - \lambda\cdot\mathrm{cost}^{(m)}(\mathbf{x})\ ] \quad (1)$$

where $\{\mathrm{acc}^{(m)}(\mathbf{x}), \mathrm{cost}^{(m)}(\mathbf{x})\}$ denote response accuracy and cost incurred by models $m\in\mathcal{M}$ for the given test query denoted by its sentence encoding $\mathbf{x}\in\mathbb{R}^{d_{\mathrm{enc}}}$, which is generated by an encoder language model [2] Cer et al. (2018). The Lagrangian parameter λ controls the relative importance the router gives to accuracy versus the cost of the generated responses.

[2]We refer to language queries through their sentence encoding Cer et al. (2018) representations $\mathbf{x}\in\mathbb{R}^{d_{\mathrm{enc}}}$ for our analysis, as it provides a fixed-dimensional representation of the abstract space of language queries.

We denote the scalar objective value $U^{(m)}(\cdot):\mathbb{R}^{d_{\mathrm{enc}}}\to\mathbb{R}$ corresponding to any model m given query $\mathbf{x}$ as

$$U^{(m)}(\mathbf{x}) = \mathrm{acc}^{(m)}(\mathbf{x}) - \lambda\cdot\mathrm{cost}^{(m)}(\mathbf{x}) \quad (2)$$

$$= \overline{U}^{(m)}(\mathbf{x}) + \epsilon^{(m)}(\mathbf{x}) \quad (3)$$

where we decompose the objective into its expected value $\overline{U}^{(m)}(\mathbf{x}) = \mathbb{E}[U^{(m)}(\mathbf{x})]$ for given query $\mathbf{x}$, and $\epsilon^{(m)}(\mathbf{x})$ a zero mean noise that captures the effect of generation randomness due to the stochastic nature of language models leading to different response correctness and response lengths. Since the stochastic noise is difficult to control or estimate, we reframe the query routing problem as follows.

$$m^*(\mathbf{x}) \leftarrow \arg\max_{m\in\mathcal{M}} \overline{U}^{(m)}(\mathbf{x}) \quad (4)$$

$$\approx \arg\max_{m\in\mathcal{M}} \widehat{U}^{(m)}(\mathbf{x}) \quad (5)$$

Thus, the main task of the router is to produce estimates $\widehat{U}^{(m)}(\mathbf{x})$ of $\overline{U}^{(m)}(\mathbf{x})$ for each incoming query $\mathbf{x}$. A mismatch between training queries and test queries leads to poor estimation of $\widehat{U}^{(m)}(\mathbf{x})$ and eventually misrouting through eq. (5).

Existing nonparametric routers such as KMeans and kNN produce these estimates using the nearest cluster of training queries or the k nearest neighboring training queries. For outlier queries that significantly differ from training queries, KMeans and kNN routers also produce poor estimates $\widehat{U}^{(m)}(\mathbf{x})$, resulting in suboptimal routing decisions. Below, we present a general framework that subsumes these existing routers, and lays the foundation for our proposed ProxRouter, which is more robust to outliers.

2.2 A Unified Representation of Clustering and Nearest Neighbors Routers

For a training set of queries $\mathcal{X}_{\mathrm{tr}}\subseteq\mathcal{X}$ in the space of all possible queries $\mathcal{X}$, we have model accuracy and cost observations from previously generated responses. The encoding space projections $\{\mathbf{x}_{\mathrm{tr}} : \mathbf{x}_{\mathrm{tr}}\in\mathcal{X}_{\mathrm{tr}}\}$ of the training queries (denoted by $\mathbf{x}_{\mathrm{tr}}$) forms the basis of all nonparametric routers. We introduce the notion of a reference set $\mathcal{I}$ which represents the sources of information used by the estimator $\widehat{U}^{(m)}(\mathbf{x})$ for any test query $\mathbf{x}$. Each element $\{i : (\mathbf{r}_i, V_i^{(m)})\}$ in the reference set is indexed by $i\in\{1,\cdots,|\mathcal{I}|\}$, and has two associated attributes: the reference vector $\mathbf{r}_i\in\mathbb{R}^{d_{\mathrm{enc}}}$ in the encoding space, and $V_i^{(m)}\in\mathbb{R}$ denoting the objective value.

For clustering-based routers (eg. KMeans router), the reference set $\mathcal{I}$ corresponds to the set of clusters ($|\mathcal{I}| = K$, number of clusters). The number of

queries in each cluster c_i for $i \in \{1, \cdots, |\mathcal{I}|\}$ is denoted by n_i. The reference vector $\mathbf{r}_i = \sum_{\mathbf{x}_{\text{tr}} \in c_i} \mathbf{x}_{\text{tr}}/n_i$ is the geometric centroid of training query encodings in the cluster c_i. The model objective value $V_i^{(m)} = \sum_{\mathbf{x}_{\text{tr}} \in c_i} U^{(m)}(\mathbf{x}_{\text{tr}})/n_i$ is the average of the observed model objective values for training queries in the cluster c_i, where $U^{(m)}(\mathbf{x}_{\text{tr}}) = \text{acc}^{(m)}(\mathbf{x}_{\text{tr}}) - \lambda \cdot \text{cost}^{(m)}(\mathbf{x}_{\text{tr}})$ for a fixed λ.

For nearest neighbor based routers (eg. kNN router), each training query $\mathbf{x}_{\text{tr}} \in \mathcal{X}_{\text{tr}}$ corresponds to unique element in reference set $\mathcal{I}$ ($|\mathcal{I}| = |\mathcal{X}_{\text{tr}}|$, the number of queries in training set). The representative vector $\mathbf{r}_i = \mathbf{x}_{\text{tr}}$ is the encoding vector for training query itself, and the model objective value $V_i^{(m)} = U^{(m)}(\mathbf{x}_{\text{tr}}) = \text{acc}^{(m)}(\mathbf{x}_{\text{tr}}) - \lambda \cdot \text{cost}^{(m)}(\mathbf{x}_{\text{tr}})$ for query $\mathbf{x}_{\text{tr}}$.

Using the reference set notation above, we denote the family of estimators of $\overline{U}^{(m)}(\mathbf{x})$ for any test query $\mathbf{x}$ as

$$\widehat{U}^{(m)}(\mathbf{x}) = \sum_{i \in [|\mathcal{I}|]} w_i(\mathbf{x}) V_i^{(m)} \tag{6}$$

where $w_i(\mathbf{x}) \geq 0$, $\sum_{i \in [|\mathcal{I}|]} w_i(\mathbf{x}) = 1$.

KMeans and kNN routers as Special Cases The baseline KMeans router allocates weight $w_i(\mathbf{x}) = 1$ for the index i corresponding to the closest cluster c^* to test query $\mathbf{x}$ and $w_i(\mathbf{x}) = 0$ to all other clusters. The kNN router allocates $w_i(\mathbf{x}) = 1/k$ for the k nearest neighbors of query $\mathbf{x}$ and $w_i(\mathbf{x}) = 0$ to all other reference points. The distance $d(\mathbf{x}, \mathbf{r}_i)$ between the test query encoding $\mathbf{x}$ and reference vectors $\mathbf{r}_i$ is calculated by an appropriate measure $d(\cdot, \cdot) : \mathbb{R}^{d_{\text{enc}}} \times \mathbb{R}^{d_{\text{enc}}} \to \mathbb{R}$ such as cosine distance, Euclidean distance etc. whose choice is guided by the design of the deterministic encoder model. We utilize cosine distance in our experimental analysis, as it is suitable for a variety of sentence encoders.

Drawbacks of Existing Techniques for Oulier Queries For KMeans router, closest cluster assignment presents a hard decision boundary, and thus, slight changes in the test query $\mathbf{x}$ lead to abrupt changes in the predicted objective values in eq. (6), especially for outlier queries that are well outside the clusters. Moreover, KMeans clustering is agnostic to intracluster spread, which is indicative of the variance $\text{Var}[V_i^{(m)}]$ of model objectives for individual reference set elements indexed by $i \in \{1, \cdots, |\mathcal{I}|\}$. In kNN routers, outlier queries whose k nearest neighbors lie far away suffer under uniform weighting as distant neighbors receive the same weight as closer ones, degrading the estimate. These issues adversely impact routing performance on outlier queries, which we seek to address via our proposed proximity-weighted router described below.

3 PROXROUTER: PROXIMITY WEIGHTED AGGREGATION ROUTER

To improve the model accuracy and cost estimates produced by the query router for outlier queries while preserving inlier performance, our proposed method ProxRouter (Algorithm 1) defines new aggregation weights in eq. (6), and applies them to all test queries (both inliers and outliers) alike, thereby eliminating any need for inlier-vs-outlier classification for a test query. It initializes the aggregation weights in eq. (6) as minimum variance priors for test query $\mathbf{x}$ (which we refer to as $\mathbf{p}(\mathbf{x}) = [p_1(\mathbf{x}), \cdots, p_{|\mathcal{I}|}(\mathbf{x})]$) over the reference set, and then exponentially tilts them for each reference element based on proximity to the query $\mathbf{x}$ to obtain bias-controlled weights $\mathbf{w}(\mathbf{x}) = [w_1(\mathbf{x}), \cdots, w_{|\mathcal{I}|}(\mathbf{x})]$. This two-stage design approach, explained in detail below, results in better estimates of the objective values $\widehat{U}^{(m)}(\mathbf{x})$ and improved routing decisions. While we focus our attention on improving the robustness of the standard KMeans and kNN-based query routers due to their widespread adoption and interpretability, more advanced techniques such as fuzzy, spectral, and kernel methods have been studied in statistics literature Ferraro (2024); Ng et al. (2001); Wand and Jones (1995); Berger (1985). Future work may consider extending this approach to more advanced statistical methods.

Algorithm 1 ProxRouter

Input: Test query $\mathbf{x}$
Given: Training queries $\mathcal{X}_{\text{tr}}$, LLM pool $\mathcal{M}$, and Reference set elements $\{i : (\mathbf{r}_i, V_i^{(m)})\}\ \forall\ i \in [|\mathcal{I}|]$.
Objective: Find $\arg\max_{m \in \mathcal{M}} U^{(m)}(\mathbf{x})$.
Procedure:
  $p_i(\mathbf{x}) \propto [\text{Var}(V_i^{(m)})]^{-1}$ (least variance priors)
  $w_i(\mathbf{x}) \propto p_i(\mathbf{x}) \exp(-\phi_i(\mathbf{x})/\tau)$ (proximity tilting)
  $\widehat{U}^{(m)}(\mathbf{x}) \leftarrow \sum_i w_i(\mathbf{x})\, V_i^{(m)}$ (estimated objective)
  $m^\star(\mathbf{x}) \leftarrow \arg\max_{m \in \mathcal{M}} \widehat{U}^{(m)}(\mathbf{x})$ (chosen best model)
Return: $m^\star(\mathbf{x})$.

3.1 Least Variance Priors

When choosing the aggregation weights $w_i(\mathbf{x})$ in the estimate $\widehat{U}^{(m)}(\mathbf{x})$ as per eq. (6), one must consider the heterogeneous variance levels $\text{Var}[V_i^{(m)}]$ for elements in the reference set. The optimal least variance estimator weights (priors) $\mathbf{p}(\mathbf{x})$ are of the form of $p_i(\mathbf{x}) \propto (\text{Var}[V_i^{(m)}])^{-1}$. The proof involves minimizing the variance of RHS in eq. (6) subject to the weights summing to unity, and it is detailed in Appendix D.

For the KMeans clustering router, the variance of each cluster summary $\text{Var}[V_i^{(m)}]$ can be expressed as the sum

of variances of the n_i training query points ($\mathbf{x}_{\text{tr}} \in c_i$) belonging to cluster c_i, and under the independence assumption (see Appendix F):

$$\begin{aligned}\text{Var}[V_i^{(m)}] &= \mathbb{E}\left[\frac{1}{n_i^2}\sum_{\mathbf{x}_{\text{tr}}\in c_i}\left(\epsilon^{(m)}(\mathbf{x}_{\text{tr}})\right)^2\right] \\ &= \frac{1}{n_i^2}\sum_{\mathbf{x}_{\text{tr}}\in c_i}\text{Var}\left[\epsilon^{(m)}(\mathbf{x}_{\text{tr}})\right]. \qquad (7)\end{aligned}$$

Since the exact per-query variance $\text{Var}[\epsilon^{(m)}(\mathbf{x}_{\text{tr}})]$ is unknown in practice, we estimate $\text{Var}[V_i^{(m)}]$ based on heuristics derived from query similarity in the encoding space. Intuitively, clusters that are more geometrically dispersed contain semantically diverse queries, implying that $\text{Var}[V_i^{(m)}]$ within such clusters is larger than in compact clusters. Formally, we define the intra-cluster "spread" as $s_i = \sum_{\mathbf{x}_{\text{tr}}\in c_i} d(\mathbf{x}_{\text{tr}}, \mathbf{r}_i)/n_i$, the mean distance of queries in c_i from their centroid $\mathbf{r}_i$. Consequently, $\text{Var}[V_i^{(m)}]$ grows with s_i but decreases with n_i (as suggested by eq. (7), where variance accumulates linearly in n_i but is normalized quadratically). Based on this intuition, we assume low-variance prior weights of the form $p_i(\mathbf{x}) \propto n_i/s_i$ for KMeans clustering routers.

For kNN routers, since it is difficult to estimate per-query variance $\text{Var}[\epsilon^{(m)}(\mathbf{x}_{\text{tr}})]$, we assume the prior probabilities $p_i(\mathbf{x}) = 1/k$ for the k reference elements that are closest to the test query $\mathbf{x}$, and zero otherwise.

3.2 Controlling the Bias-Variance trade-off via Proximity-Based Tilting

The minimum-variance estimator, denoted by $\mathbf{p}(\mathbf{x})$ in Section 3.1, suffers from high bias because it does not adequately prioritize reference elements that are close to the test query $\mathbf{x}$. To mitigate this bias, we introduce an exponentially tilted aggregation that reweights the minimum-variance estimates $\mathbf{p}(\mathbf{x})$ using a proximity penalty $\phi_i(\mathbf{x})$, which is a monotonically increasing function of the distance $d(\mathbf{x}, \mathbf{r}_i)$ to the reference element indexed by i. Formally, we transform the low-variance priors $\mathbf{p}(\mathbf{x})$ into new intensities $\theta_i(\mathbf{x}) \propto p_i(\mathbf{x})\,\exp(-\phi_i(\mathbf{x})/\tau)$, and normalize to obtain the proximity aware weights $w_i(\mathbf{x}) = \theta_i(\mathbf{x})/\sum_{i\in[|\mathcal{I}|]}\theta_i(\mathbf{x})$.

Conceptually, the above proximity-aware weights are the solution to the following optimization problem in eq. (8), which trades off between reducing bias (corresponding to the first term) and reducing variance (corresponding to the second term). The proof details are given in Appendix E.

$$\min_{\mathbf{w}(\cdot)\in\Delta^{|\mathcal{I}|}} \sum_{i\in[|\mathcal{I}|]} w_i(\mathbf{x})\,\phi_i(\mathbf{x}) + \tau\, D_{\text{KL}}(\mathbf{w}(\mathbf{x})\|\mathbf{p}(\mathbf{x})) \quad (8)$$
$$\Rightarrow w_i(\mathbf{x}) \propto p_i(\mathbf{x})\, e^{-\phi_i(\mathbf{x})/\tau}.$$

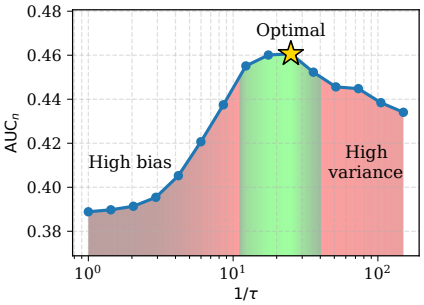


Figure 4: Bias-variance tradeoff governed by proximity based prioritization (see Section 4 for details). Increasing $1/\tau$ strengthens proximity weighting and raises variance (and vice versa for smaller $1/\tau$). Routing performance is measured as area under the mean accuracy-cost curve (AUC), normalized by cost range.

This framework provides a controllable bias–variance tradeoff: increasing τ drives the weights toward the low-variance priors $\mathbf{p}(\mathbf{x})$, while decreasing τ emphasizes proximity, thereby reducing bias by preferring semantically similar reference elements. Notably, existing routing schemes emerge as special cases:

- KMeans router with closest cluster assignment is equivalent to $\tau = 0$ (no KL penalty).
- kNN router with uniform averaging over neighbors is equivalent to $\tau \to \infty$ with uniform priors $\mathbf{p}(\mathbf{x})$ over the k nearest neighbors.

Figure 4 illustrates bias-variance tradeoff, where varying τ interpolates between high-bias (small $1/\tau$) and high-variance (large $1/\tau$) regimes, with optimal router performance typically achieved at an intermediate value $1/\tau^*$ that balances proximity confidence against variance reduction. In practice, τ is tuned on held-out data or adapted to the scale of the distance metric.

Notably, *ProxRouter preserves the original clustering or nearest-neighbors router design.* It is lightweight: the only additional computation involves evaluating low-variance priors and applying the proximity-aware tilt (Appendix C). Moreover, the framework is flexible: it can incorporate a wide range of distance measures in the encoding space, such as Euclidean distance or cosine dissimilarity. Overall, ProxRouter consistently improves routing by (i) lowering variance through principled priors and (ii) reducing bias via proximity-aware weighting.

4 EXPERIMENTAL ANALYSIS

Guided by the proximity-aware aggregation described in Section 3, we design and examine the performance

Table 1: Average Accuracies of the models in our considered pool on query datasets used for our experiments. Best two models for each dataset are emphasized in shades of green, the worst two models for that dataset are shown in shades of red. Numbers shown for explicit zero shot prompting (refer Appendix B.2).

Model \Dataset	SVAMP	GSM8k	HSwag	SciQ	SoQA	WinoG	MedQ	CSQA	LogiQ	BBH	Avg.
Qwen2-1.5B	23.29	68.05	30.60	56.15	51.05	54.30	**22.10**	41.95	31.20	**44.80**	42.35
Qwen2-7B	51.43	29.85	77.30	90.80	82.05	73.10	48.50	77.55	53.60	72.80	65.70
Deepseek-math-7B	67.86	**77.05**	35.20	68.05	61.35	52.40	24.20	44.30	35.70	57.60	52.37
Gemma-2B	22.57	**4.05**	31.80	55.50	50.25	**50.90**	23.50	36.85	30.00	56.80	36.22
Gemma-7B	52.71	13.10	50.75	81.95	70.95	57.60	32.50	62.05	38.60	64.80	52.50
Phi-3-small-7B	65.43	17.55	79.20	94.10	82.05	**85.30**	58.40	75.50	54.70	76.00	68.82
Llama-2-7B	39.14	7.00	47.60	76.50	63.85	51.90	26.10	49.95	38.00	**44.80**	44.48
Llama-2-13B	47.29	12.10	57.75	80.65	67.90	55.60	32.50	55.95	40.40	54.40	50.45
Llama-3.2-1B	**14.14**	4.50	**27.35**	**47.40**	**40.60**	51.70	22.80	**25.55**	**26.05**	51.20	**31.13**
Llama-3.2-3B	35.29	12.90	44.40	91.75	70.85	53.40	46.80	65.85	38.25	60.00	51.95
Llama-3.1-8B	37.14	44.10	53.00	91.50	72.95	58.10	46.00	68.35	43.00	64.80	57.89
Llama-3.3-70B	**86.29**	40.80	**87.05**	**97.65**	**83.40**	83.80	**84.70**	**80.45**	**60.45**	79.20	**78.38**
Mistral-7B	52.00	22.55	69.55	85.55	74.50	59.30	45.40	68.15	45.85	74.40	59.72
Mixtral-8x7B	59.57	26.15	74.10	87.05	71.85	68.60	53.90	67.65	50.35	**82.40**	64.16
Max across models	86.29	77.05	87.05	97.65	83.40	85.30	84.70	80.45	60.45	82.40	78.38
Mean across models	46.72	27.12	54.69	78.90	67.40	61.14	40.53	58.58	41.87	63.14	54.01
Min across models	14.14	4.05	27.35	47.40	40.60	50.90	22.10	25.55	26.05	44.80	31.13

of ProxRouter for both clustering based (KMeans) and nearest neighbor (kNN) settings using corresponding choices of the reference set and its elements. As remarked earlier, a central challenge in router performance is generalization to queries from tasks that are absent or underrepresented in the training query set $\{\mathbf{x}_{\text{tr}} : \mathbf{x}_{\text{tr}} \in \mathcal{X}_{\text{tr}}\}$. In the subsequent assessment, we evaluate ProxRouter and corresponding baseline with the same training query set $\mathcal{X}_{\text{tr}}$, and contextualize the improvement in performance with respect to the full knowledge training query set, modeling $\mathcal{X}_{\text{tr}} \approx \mathcal{X}$ for an upper bound on routing performance.

We first describe our experimental setup and evaluation regimes (Section 4.1), and then describe the performance of ProxRouter (in both KMeans and kNN setting) on a variety of tasks (Section 4.2) and comment on router retraining.

4.1 Experimental and Evaluation Setup

We conduct our study on ten publicly available language query datasets, choosing multiple inlier/outlier task configurations. Our model pool comprises of 14 diverse LLMs covering a wide spectrum of parameter count and capabilities. A summary of the query datasets and models, along with average accuracies for each model–dataset pair, appears in Table 1. For query encodings, we employ the MPNet-base sentence encoder Song et al. (2020). Additional details such as dataset characteristics, LLM attributes, prompting strategies, inference costs, and the sentence-encoder selection are provided in Appendix B.

We identify two sources of under representation of queries in the training set $\mathcal{X}_{\text{tr}}$: Leave-Task-Out scenario where no training queries represent outlier tasks, and Few-Shot-Outlier scenario where very few queries from outlier tasks are present in training set. Leave-Task-Out scenario naturally arises in clustering based routers where no unique cluster represents outlier queries, whereas Few-Shot-Outlier case is typical to nearest neighbor routers where uniform averaging over neighbors obscures information conveyed by few training queries that are closer to outlier test query.

For clustering based (KMeans) query routers, we evaluate our approach KM-Prox with the closest cluster assignment policy KM-Base in the Leave-Task-Out regime. For nearest neighbor based (kNN) query routers, we evaluate our approach kNN-Prox with the uniform nearest neighbor averaging router kNN-Base in the Few-Shot-Outlier scenario. Additionally, we also interpret improvement in outlier generalization of KM-Prox (kNN-Prox) with KM-AllSee (kNN-AllSee) respectively which have seen all queries in training set, modeling the full knowledge case $\mathcal{X}_{\text{tr}} \approx \mathcal{X}$ and providing an upper bound in routing performance. We also include two router-agnostic baselines: *(i)* Expensive Model routing to the biggest generic model in the pool (Llama 70B in our case), *(ii)* Random Routing to any model from the pool.

A preferable router is one which yields higher accuracy values on test queries at the same cost, or alternatively lower costs while maintaining same accuracy level. To quantify router performance based on this observation, we use area under the mean accuracy cost plot normalized by the cost range Hu et al. (2024); Jitkrittum et al. (2025a), a value in $[0, 1]$ which we denote by a percentage scale.

4.2 Results

KM-Prox Across all KM-Base, KM-Prox and KM-AllSee, we set number of clusters $K = 32$, proximity penalty $\phi_i(\mathbf{x}) = d(\mathbf{x}, \mathbf{r}_i)$ as the cosine distance between test query and clusters, and $1/\tau = 20$.

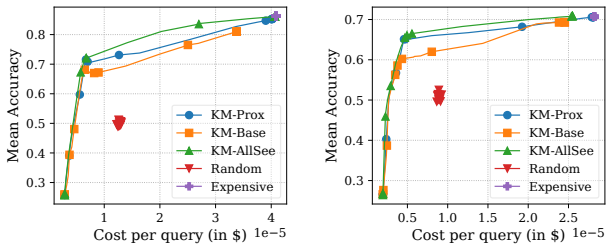


Figure 5: KM-Prox, KM-Base and KM-AllSee router performance on *(left)* MedQA, HellaSwag as outlier tasks and *(right)* LogiQA, CommonSenseQA, BBH-BoolEx as outlier tasks. KM-Prox consistently improves routing performance over KM-Base without additional training data.

We consider two sets of task distributions for our analysis of ProxRouter on clustering based settings following the Leave-Task-Out generalization, *(i)* Hellaswag and MedQA as outliers and *(ii)* LogiQA, BigBenchHard (BoolEx), CommonsenseQA as outliers. In both cases, outlier tasks are absent in $\mathcal{X}_{\text{tr}}$, and all other tasks from Table 1 are present as both training queries and test queries. We split the inlier tasks into a 60-40 train-test split, and outlier tasks fully present in the test split.

Table 2: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for chosen sets of outlier tasks. Upper bound denoted by the full knowledge router KM-AllSee. ProxRouter improves generalization on outlier queries, with unaffected routing performance on inlier queries across different outlier configurations.

Outlier Tasks	Split	Routing method		Upper Bound
		KM-Base	KM-Prox	KM-AllSee
Hellaswag, MedQA	Outlier	70.68%	74.88%	78.36%
	Inlier	74.62%	74.86%	74.63%
	Overall	73.04%	75.12%	74.87%
LogiQA, CSQA, BBH	Outlier	63.39%	66.18%	67.25%
	Inlier	79.35%	79.92%	79.74%
	Overall	71.61%	73.46%	73.88%

We observe that for both outlier cases (Figure 5), KM-Prox improves upon the normalized AUC of the mean accuracy cost plot as compared to the KM-Base with the same $\mathcal{X}_{\text{tr}}$ at practically no additional computational costs for routing (Appendix C), and bridges the gap between the KM-AllSee full knowledge router upper bound. Notably our approach provides not just overall higher AUC, but provides better accuracy at almost all operating cost regions in the mean accuracy cost plot. Additionally we highlight that all three (KM-Base, KM-Prox, KM-AllSee) perform identically for inlier tasks, indicating unaffected inlier performance of ProxRouter with better generalizability to unseen outlier tasks. Supplementary results in Appendix H.

kNN-Prox Across kNN-Base, kNN-Prox and kNN-AllSee, we set number of neighbors $k = 100$, proximity penalty $\phi_i(\mathbf{x}) = d(\mathbf{x}, \mathbf{r}_i)$ as the cosine distance between test query and its neighbors, and $1/\tau = 20$. We consider the Few-shot Outlier case for evaluating ProxRouter in the nearest neighbor based setting where $\mathcal{X}_{\text{tr}}$ comprises of few (~ 25) queries from math tasks (GSM8k, SVAMP).

Table 3: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for chosen outlier tasks. Upper bound denoted by the full knowledge router kNN-AllSee.

Outlier Tasks	Split	Routing method		Upper Bound
		kNN-Base	kNN-Prox	kNN-AllSee
GSM8k, SVAMP	Outlier	38.55%	46.64%	60.77%
	Inlier	77.51%	76.96%	79.11%
	Overall	63.98%	68.12%	74.60%

In accordance with previous observations, ProxRouter significantly improves outlier routing performance (Table 3), increasing the AUC_n by 8.1 pp (38.5% to 46.6%). Remarkably, kNN-Prox achieves a higher accuracy value at a significantly lower average cost than kNN-Base due to the presence of fine-tuned models that outperform largest model, as depicted in Figure 6. Figure 7 describes the matching accuracy of routing through kNN-Base, kNN-Prox and kNN-AllSee with post-hoc top-$(1, 3, 5)$ suitable models as per objective in eq. (1), indicating that kNN-Prox significantly improves routing performance as compared to kNN-Base with the same training query set.

Extreme OOD Testing In real world scenarios, heavy distribution shifts may arise leading to substantial difference between training and test tasks. To capture such scenario in our setup, we restrict analysis to Winogrande (commonsense questions), MedQA (clinical questions), and Math (GSM8K + SVAMP), whose queries are semantically very distinct and far apart in embedding space (fig. 2). In this setup, we treat MedQA as the outlier task and report results in attached table 4. ProxRouter achieves a normalized AUC on outlier queries of 66.6%, compared to 53.98% for the Base router, while the full-knowledge AllSee upper bound is 70.27%; inlier performance is almost identical across all three. We emphasize that the improvement is not uniform across all choices of inlier and outlier tasks, and that some extreme OOD configurations remain very challenging. We present ProxRouter as a practical method for improving robustness under realistic domain shifts, rather than a worst-case guarantee for arbitrarily OOD scenarios.

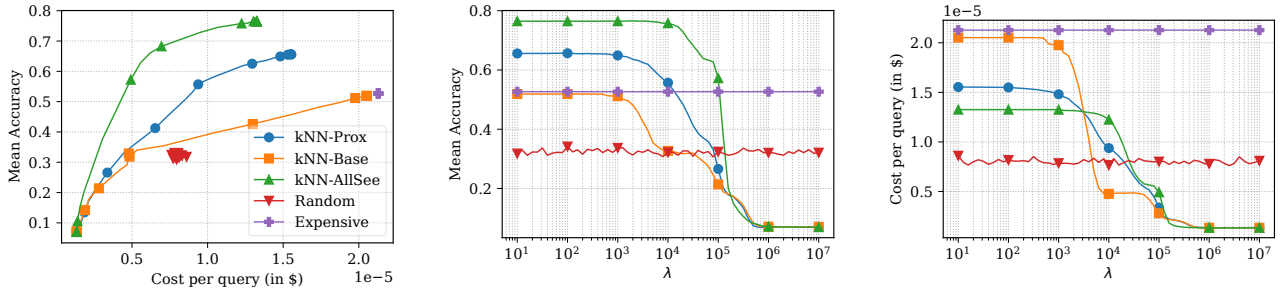


Figure 6: kNN-Prox, kNN-Base and kNN-AllSee router performance on (GSM8k, SVAMP) outliers. *Left* plot describes the mean accuracy-cost plot for outlier queries. *Center* plot describes average accuracy vs λ and *right* plot describes average cost per query vs λ (eq. (1)). kNN-Prox greatly improves robustness and generalization to outlier queries by proximity-based prioritization of queries in the training set.

Table 4: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base to simulate extreme outlier scenario and examine impact of bias and variance reduction separately. Queries are sampled from Math Tasks (SVAMP + GSM8K), Winogrande (fill in the blanks commonsense reasoning), MedQA (factual recall). Each of these tasks represent very different abilities of LLMs. MedQA is chosen as the outlier task. *Min-Var:* use only the least variance priors. *Tilt-Uniform:* perform proximity tilting, but with uniform priors instead of min-var. *Uniform:* uniformly average over all clusters.

Split	Routing method					Upper Bound
	KM-Base	Min-Var	Tilt-Uniform	Uniform	KM-Prox	KM-AllSee
Outlier	53.98%	50.79%	41.13%	33.99%	66.60%	70.27%
Inlier	64.17%	56.76%	62.19%	53.69%	62.99%	63.76%
Overall	61.23%	54.99%	57.19%	43.54%	67.45%	70.29%

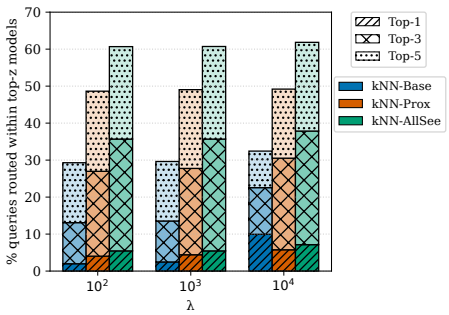


Figure 7: Routing Match Accuracies for kNN-Prox, kNN-Base, kNN-AllSee measured against post-hoc top-(1,3,5) most suitable models (by eq. (1)). Higher is better.

Impact of Bias and Variance Reduction To examine the roles of the minimum-variance prior and subsequent bias reduction in ProxRouter, we compare against three variants: *(i)* **Uniform** weights over reference-set elements, *(ii)* **Min-Var**, which uses the minimum-variance priors directly as weights (variance reduction only), and *(iii)* **Tilt-Uniform**, which applies proximity tilting starting from a uniform prior (bias reduction only).

We use the same extreme OOD configuration described above, where Base, ProxRouter, and AllSee exhibit large performance gaps, enabling us to more clearly "peel off" the effects of variance and bias reduction. Empirically (attached table 4), **Uniform** performs worst; **Min-Var** reduces variance and improves over Uniform but suffers from high bias; **Tilt-Uniform** better controls bias but suffers from higher variance (still outperforming Uniform); and combining both components yields ProxRouter, which performs best on both inlier and outlier queries. The relative importance of bias versus variance reduction varies across experimental setups, hence we instead emphasize the benefit of their combination.

Outlier gaps and Router Retraining We observe that the performance gap between the AllSee and the Base router varies with the choice of the outlier task under examination (Figures 5 and 6). These differences may arise because some outlier tasks may demand LLM abilities that co-occur with abilities required for inlier tasks, leading to small gaps as the router already has some idea about handling such outliers, while completely new tasks that require very different LLM skill sets incur larger drops as router has not mapped such LLM abilities in the training query set. To quantify this effect, we measure the similarity of model rankings between inliers and outliers using a simple overlap metric. For any task t and λ value, let $S_z(t, \lambda)$ be the set of top-z models by ranked objective value (eq. (1)). For each outlier task t_{out} and inlier task t_{in}, we define

top-z Jaccard similarity

$$J_z(t_{\text{out}}, t_{\text{in}}, \lambda) = \frac{|S_z(t_{\text{out}}, \lambda) \cap S_z(t_{\text{in}}, \lambda)|}{|S_z(t_{\text{out}}, \lambda) \cup S_z(t_{\text{in}}, \lambda)|},$$

and average across all $(t_{\text{out}}, t_{\text{in}})$ pairs for overall J_z^λ.

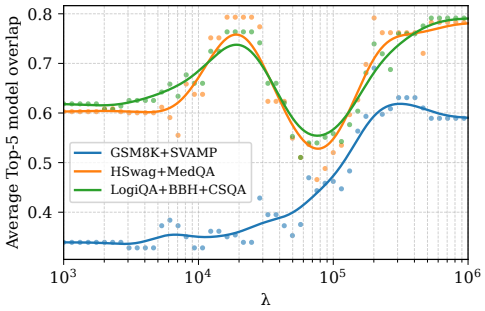


Figure 8: Jaccard overlap of top-5 models for three different outlier task sets with corresponding inliers (keeping all but outlier tasks as inliers in the training query set, for each of the three sets). Math tasks (GSM8k, SVAMP) with smaller top-5 model overlap with inlier tasks exhibit larger difference in AllSee and Base router (Figure 6 and Table 3).

This measure captures why the Base-AllSee gaps differ, as low average J_z^λ value indicates that outliers favor very different models than inlier tasks, leading to larger Base-AllSee gaps. Alternatively, higher J_z^λ indicates shared model preferences between inliers and outliers, and consequently smaller gaps. Math outliers (GSM8k+SVAMP) have low top-z overlap with inliers (Figure 8), whereas the other outlier sets have higher overlap. Higher overlap likely reflects co-existing skills in language models (e.g., reasoning with logical QA, medical with general knowledge); lower overlap reflects a lack of shared skills. This overlap metric also informs *when to retrain* the router, as consistently low average J_z^λ for new test queries might be indicative of test-train mismatch of new queries for existing router. Router retraining through evaluation of queries on all models can be triggered when J_z^λ falls below a validation chosen threshold for incoming test queries. The choice of threshold can be determined by a cost-aware policy which considers the cost of evaluating outlier queries on all models against the drop in routing performance of existing router. ProxRouter increases robustness between such retraining periods, allowing for a reliable generalization without hindering inlier performance.

5 CONCLUSION

In this work, we address the practical challenge of train–test mismatch in language query routing. We present ProxRouter, a proximity-weighted framework for nonparametric routers that improves robustness to outlier queries while preserving inlier performance. The key idea is to prioritize nearby training queries when estimating each model's accuracy–cost values for a test query. By exponentially tilting model accuracy-cost estimates, ProxRouter controls the bias-variance tradeoff with a single hyperparameter. We also formalize a broad family of nonparametric language query routers, providing a unified lens for analysis and implementation. Extensive experiments support these claims and underscore ProxRouter's practical value. We also describe how router retraining decisions can be made by judging the similarity of model rankings on test queries with inlier queries. While this work focuses on nearest neighbours (kNN) and clustering (KMeans) based routers, exploring connections to advanced statistical techniques remains a promising direction.

Acknowledgments

This work was partially supported by NSF grants CCF 2045694, CNS-2112471, CPS-2111751, CCF-2428569, ONR grant N00014-23-1-2149, and an AI2C Seed grant. This work used Bridges-2 GPU at the Pittsburgh Supercomputing Center through allocation CIS250429 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by NSF grants #2138259, #2138286, #2138307, #2137603, and #2138296 (Boerner et al., 2023).

References

Marah Abdin and et al. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models, 2025. URL https://arxiv.org/abs/2310.12963.

Artificial Analysis. Artificial analysis: Ai model & api providers analysis. https://artificialanalysis.ai, 2025. Accessed: 2025-07-15.

James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 2nd edition, 1985. ISBN 0387960988.

Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. Access: Advancing innovation: Nsf's advanced cyberinfrastructure coordination ecosystem: Services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, PEARC '23, page 173–176, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399852. doi: 10.1145/3569951.3597559. URL https://doi.org/10.1145/3569951.3597559.

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. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018. URL https://arxiv.org/abs/1803.11175.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models, 2024. URL https://arxiv.org/abs/2409.19886.

Zhou Chen, Zhiqiang Wei, Yuqi Bai, Xue Xiong, and Jianmin Wu. Tagrouter: Learning route to llms through tags for open-domain text generation tasks, 2025. URL https://arxiv.org/abs/2506.12473.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995. doi: 10.1109/34.400568.

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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.

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

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute, 2025. URL https://arxiv.org/abs/2506.22716.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Sohl-dickstein, Kevin Murphy, and Charles Sutton. Language model cascades, 2022. URL https://arxiv.org/abs/2207.10342.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2025. URL https://arxiv.org/abs/2401.08281.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections, 2025. URL https://arxiv.org/abs/2410.03834.

Maria Brigida Ferraro. Fuzzy k-means: history and applications. *Econometrics and Statistics*, 30:110–123, 2024. ISSN 2452-3062. doi: https://doi.org/10.1016/j.ecosta.2021.11.008. URL https://www.sciencedirect.com/science/article/pii/S2452306221001398.

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

Evan Frick, Connor Chen, Joseph Tennyson, Tianle Li, Wei-Lin Chiang, Anastasios N. Angelopoulos, and Ion Stoica. Prompt-to-leaderboard, 2025. URL https://arxiv.org/abs/2502.14855.

Aaron Grattafiori and et al. 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 Ré. Smoothie: Label free language model routing, 2024. URL https://arxiv.org/abs/2412.04692.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

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

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-LLM routing system. In *Agentic Markets Workshop at ICML 2024*, 2024. URL https://openreview.net/forum?id=IVXmV8Uxwh.

Zhongzhan Huang, Guoming Ling, 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.

Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2022.11.139. URL https://www.sciencedirect.com/science/article/pii/S0020025522014633.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021.

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

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal LLM routing with correctness-based representation. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2025b. URL https://openreview.net/forum?id=QpOCijgaBE.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention, 2023. URL https://arxiv.org/abs/2309.06180.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

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

Yang Li. Rethinking predictive modeling for llm routing: When simple knn beats complex learned routers, 2025b. URL https://arxiv.org/abs/2505.12601.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning, 2020. URL https://arxiv.org/abs/2007.08124.

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, 2023. URL https://arxiv.org/abs/2311.08692.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

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

NVIDIA Corporation. Nvidia h100 pcie gpu — product brief. Technical Report PB-11133-001, NVIDIA, September 2022. URL https://www.nvidia.com/content/dam/en-zz/Solutions/gtcs22/data-center/h100/PB-11133-001_v01.pdf.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms with preference data, 2025. URL https://arxiv.org/abs/2406.18665.

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

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Manhin Poon, XiangXiang Dai, Xutong Liu, Fang Kong, John C. S. Lui, and Jinhang Zuo. Online multi-llm selection via contextual bandits under unstructured context evolution, 2025. URL https://arxiv.org/abs/2506.17670.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Marija Sakota, 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. ACM, March 2024. doi: 10.1145/3616855.3635825. URL http://dx.doi.org/10.1145/3616855.3635825.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions, 2019. URL https://arxiv.org/abs/1904.09728.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding, 2020. URL https://arxiv.org/abs/2004.09297.

Wei Song, Zhenya Huang, Cheng Cheng, Weibo Gao, Bihan Xu, GuanHao Zhao, Fei Wang, and Runze Wu. Irt-router: Effective and interpretable multi-llm routing via item response theory, 2025. URL https://arxiv.org/abs/2506.01048.

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

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Gemma Team. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

Hugo Touvron and et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

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. URL https://arxiv.org/abs/2302.13971.

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

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

M. P. Wand and M. C. Jones. *Kernel Smoothing*. Chapman & Hall, London, 1995. ISBN 0412552701. URL https://doi.org/10.1201/b14876.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions, 2017. URL https://arxiv.org/abs/1707.06209.

An Yang and et al. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Xuechen Zhang, Zijian Huang, Ege Onur Taga, Carlee Joe-Wong, Samet Oymak, and Jiasi Chen. Treacle: Thrifty reasoning via context-aware llm and prompt selection. *CoRR*, abs/2404.13082, 2024. URL https://doi.org/10.48550/arXiv.2404.13082.

Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. Beyond gpt-5: Making llms cheaper and better via performance-efficiency optimized routing, 2025a. URL https://arxiv.org/abs/2508.12631.

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

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. Eagle: Efficient training-free router for multi-llm inference, 2024. URL https://arxiv.org/abs/2409.15518.

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.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix for "ProxRouter: Proximity-Weighted LLM Query Routing for Improved Robustness to Outliers"

Appendix Index

A Related Works

The language query routing problem is concerned with selecting the appropriate model to generate responses to a query. Alternate thrusts towards efficient usage of large versus small language models additionally focus on intertwining autoregressive generations from large and small models towards the same response, or collecting multiple responses from different models. Speculative decoding Leviathan et al. (2023) aims to accelerate generation by using smaller models to predict future tokens and larger models to verify them. Multiresponse generation techniques include redundancy exploitation through best-of-n multisampling Ding et al. (2025), or through model cascading Dohan et al. (2022) where response from each model in the sequence determines what model to route to in the next round or to return the response at the current step Dekoninck et al. (2025); Chen et al. (2023). Preference aligned routers match human rating of generated responses from different models Ong et al. (2025); Frick et al. (2025) evaluated on ChatbotArena Chiang et al. (2024). Ding et al. (2024); Ong et al. (2025) consider routing across two language models, providing fundamental insights extendable to multiple

model setting. Bandit and decision making informed router design initiates dynamic inference Li (2025a) Zhang et al. (2024) Poon et al. (2025), improving performance on queries under budget constraints. Graphical models over LLMs and their outputs represent contextual information and present a weak supervised approach Guha et al. (2024); Feng et al. (2025). Jitkrittum et al. (2025a) accentuates the adaptability to a dynamic LLM pool, further highlighted by Zhang et al. (2025b) through model capability profiling. Model ability comparison through Elo scores Zhao et al. (2024) and Item Response Theory Song et al. (2025) presents an interpretable approach, furthering the understanding of router behavior. Other techniques involve utilizing query tagging among tasks Chen et al. (2025), distilling reward models Lu et al. (2023), POMDP modeling Aggarwal et al. (2025), introducing contrastive loss Chen et al. (2024) etc.

Parametric and Nonparametric Routers Structurally, most query routing approaches utilize fixed dimensional query encodings generated through deterministic encoder models Cer et al. (2018), Reimers and Gurevych (2019) which forms the backbone shared by both learned (parametric) routers and learning-free (nonparametric) routers. Building above these encodings, parametric approaches learn small multi-layer perceptrons to predict model correctness or cost Hu et al. (2024); Ding et al. (2024, 2025); Zhuang et al. (2024); Sakota et al. (2024); Huang et al. (2025). Additionally, parametric approaches also examine Bradley Terry ranking, matrix factorization and multihead classifiers Ong et al. (2025), or train smaller language model for reward modeling, task identification or hardness determination Zhang et al. (2024); Chen et al. (2025). Nonparametric routers work directly on the embedding space of training queries without learning additional parameters for making routing decisions. To canonical types are prevalent: clustering based (eg. KMeans) routers Zhang et al. (2025b,a); Jitkrittum et al. (2025b); Hu et al. (2024) and nearest neighbor based (eg. kNN) routers Li (2025b); Zhuang et al. (2024); Jitkrittum et al. (2025a); Stripelis et al. (2024). Empirical findings describe nonparametric router performance as almost equal or even better than parametric counterparts. The surprising competitiveness of simple approaches likely stems from the richness of query encoding representations generated through encoder models (fig. 2), which greatly simplifies query to model ability mapping. Furthermore, clustering and nearest neighbor routers are incremental by design, signifying that newer queries and language models can be easily added to the design without having to retrain the router as in parametric approaches. For these reasons, we adopt and focus on the nonparametric router paradigm in this work.

Robust Routing A unifying observation across multiple techniques is that query routing hinges on accurate predictions of language model correctness across a range of queries. An ideal router makes accurate routing decisions across models on all types of queries, but such a router is infeasible in practice (section 1). Prior works often quantify outlier performance drop with their approaches, but seldom aim to explicitly improve robustness and generalizability to queries from unseen tasks. We build on these observations and use theoretical insights to unify the analysis of a broad category of nonparametric routers, and utilize proximity aware aggregation coupled with variance reduction to generalize routing. We also underscore the scope of our work as being designing robust model accuracy and cost estimators, and our work can be readily plugged into multiple strategies such as model cascading, best-of-n, dynamic routing etc.

Scope and Methodological Focus A broad toolbox of existing statistical approaches can be used to model the problem of language query routing, including but not limited to fuzzy/soft KMeans Ferraro (2024), spectral clustering Ng et al. (2001), kernel smoothing Wand and Jones (1995), mode seeking methods Cheng (1995), Bayesian estimators Berger (1985) etc. These techniques have not been systematically explored for language query routing or widely adopted to our knowledge. Accordingly, we deliberately focus our study on simplistic KMeans Ikotun et al. (2023) and kNN Fix and Hodges (1989) based query routers as they are dominantly used across query routing literature and offer simplicity and interpretability. We contribute by building on these primitive yet well performing and well studied algorithms by improving their robustness and generalizability. Our unified representation for non parametric routers (section 2.2) is compatible with most of the aforementioned statistical techniques, and we highlight their examination as a promising future direction.

B Experimental Setup

Our query datasets evaluate language model abilities across multiple categories: **CommonSenseQA** Talmor et al. (2019), **GSM8K** Cobbe et al. (2021), **Hellaswag** Zellers et al. (2019), **LogiQA**Liu et al. (2020), **MedQA** Jin et al. (2021), **SciQ** Welbl et al. (2017), **Social_i_QA** Sap et al. (2019), **SVAMP** Patel et al. (2021), **Winogrande** Sakaguchi et al. (2019), **BBH** (BoolEx) Suzgun et al. (2022). Detailed prompting techniques are elaborated later in the section. Each dataset comprises of hundreds to thousands of queries. To ensure a balanced comparison across tasks, we select a subset of each for training and evaluation, as described in the Appendix B.3. We intentionally exclude multi-subtask datasets such as ARC-Challenge Clark et al. (2018) and MMLU Hendrycks et al. (2021) from our analysis in order to benchmark generalizability to unseen tasks. Including such datasets risks inadvertent data leakage, since they encompass queries from numerous subtasks (e.g., MMLU includes four distinct math-related subtasks among its 57), potentially overlapping with the test distribution and compromising evaluation integrity.

We choose language models across different capability ranges as indicated by their number of parameters: *(i)* $<$**3B** Params: Llama-3.2-1B-Instruct Grattafiori and et al (2024), Qwen2-1.5B-Instruct Yang and et al (2024), Gemma-2b-it Team (2024), Llama-3.2-3B-Instruct Grattafiori and et al (2024) *(ii)* **7B** Params: Phi-3-small-8k-instruct Abdin and et al (2024), Qwen2-7B-Instruct Yang and et al (2024), Deepseek-math-7b-instruct Shao et al. (2024), Llama-2-7b-chat-hf Touvron and et al (2023), Gemma-7b-it Team (2024), Mistral-7B-Instruct-v0.3 Jiang et al. (2023), Llama-3.1-8B-Instruct Grattafiori and et al (2024) *(iii)* **13B** Params: Llama-2-13b-chat-hf Touvron and et al (2023) *(iv)* $>$**50B** Params: Mixtral-8x7B-Instruct-v0.1 Jiang et al. (2024), Llama-3.3-70B-Instruct Grattafiori and et al (2024). Our LLM ensemble consists of standard autoregressive models, chat and instruct fine-tuned models, mixture-of-experts based models and task-specific finetuned models. The inference costs for chosen language models is provided in Appendix B.1 in \$/M tokens.

Our analysis considers query embeddings generated from the following language encoder models: MiniLM L6 Wang et al. (2020), Paraphrase Albert small Reimers and Gurevych (2019), MPNet base Song et al. (2020), DistillRoberta Sanh et al. (2019) with fixed embedding dimensions being 384, 768, 768, 768 respectively. We observe that router performance is relatively unaffected by the choice of the underlying sentence encoder model (Table 5). For the sake of uniformity, we utilize MPNet Base for our experiments.

Table 5: Base Router performance by encoder model (normalized AUC for mean acc-cost plot) $K = 32$ for KMeans router, nearest neighbours $k = 100$ for kNN router. Notably, routing performance is relatively unnaffected by the choice of encoder model. We utilitize MPNet Base encoder throughout our experiments.

Encoder	**Enc Dim**	K**Means**	k**NN**
MiniLM L6	384	76.3%	75.7%
Para Albert Small	768	77.1%	75.9%
MPNet Base	768	76.7%	75.9%
DistillRoberta	768	77.2%	75.7%

B.1 Pricing Structure for Language Models

Table 6 denotes the pricing structure of utilized language models in terms of unit input and output token prices Analysis (2025).

B.2 Prompting Techniques

We utilize zero shot prompting technique for queries from all tasks, including math queries which elicit generated response with final numeric answer in the output sequence and multiple choice questions which only require letter options corresponding to the correct answer. All models share the same evaluation approach for a given task, depending on the type and cardinality of the set of responses. We host and evaluate all opensource models locally on an array of four NVIDIA Tesla V100s (32GB) NVIDIA Corporation (2018) and four H100s (80GB) NVIDIA Corporation (2022). We use vLLM inference machine Kwon et al. (2023) for generating responses for provided

Table 6: Inference pricing in USD per 1M tokens.

Model	Input ($/1M tok)	Output ($/1M tok)
Qwen/Qwen2-1.5B-Instruct	0.015	0.030
Qwen/Qwen2-7B-Instruct	0.040	0.080
deepseek-ai/deepseek-math-7b-instruct	0.040	0.080
google/gemma-2b-it	0.015	0.030
google/gemma-7b-it	0.030	0.050
meta-llama/Llama-2-13b-chat-hf	0.050	0.100
meta-llama/Llama-2-7b-chat-hf	0.020	0.040
meta-llama/Llama-3.1-8B-Instruct	0.030	0.050
meta-llama/Llama-3.2-1B-Instruct	0.010	0.020
meta-llama/Llama-3.2-3B-Instruct	0.015	0.030
meta-llama/Llama-3.3-70B-Instruct	0.150	0.300
microsoft/Phi-3-small-8k-instruct	0.025	0.050
mistralai/Mistral-7B-Instruct-v0.3	0.020	0.050
mistralai/Mixtral-8x7B-Instruct-v0.1	0.100	0.250

prompts for the language models, with default suggested LLM hyperparameter settings and chat templates wherever applicable.

B.2.1 GSM8K

Solve the following math problem and provide a numerical answer.
Question: Jen and Tyler are gymnasts practicing flips. Jen is practicing the triple-flip while Tyler is practicing the double-flip. Jen did sixteen triple-flips during practice. Tyler flipped in the air half the number of times Jen did. How many double-flips did Tyler do?
Give only the final answer without explanation.
Answer:

B.2.2 HellaSwag

Find the most likely ending of the given question.
Question: A woman is wearing a white robe and a black belt. She does karate moves in her room. she
Choices:
Option (A): gets into a cage in the middle of her room.
Option (B): kicks her legs up several times.
Option (C): throws an object off in the distance.
Option (D): watches tv and eats ice cream on the couch.
Return the correct option letter in parenthesis () without any explanation:

B.2.3 MedQA

Please answer with the correct option letter.
Question: A 46-year-old man comes to the physician because of a 2-month history of hoarseness and drooling. Initially, he had difficulty swallowing solid food, but now he has difficulty swallowing foods like oatmeal as well. During this period, he also developed weakness in both arms and has had an 8.2 kg (18 lb) weight loss. He appears ill. His vital signs are within normal limits. Examination shows tongue atrophy and pooled oral secretions. There is diffuse muscle atrophy in all extremities. Deep tendon reflexes are 3+ in all extremities. Sensation to pinprick, light touch, and vibration is intact. An esophagogastroduodenoscopy shows no abnormalities. Which of the following is the most likely cause of this patient's symptoms?
Choices:
(A) Multiple cerebral infarctions
(B) Autoimmune destruction of acetylcholine receptors
(C) Demyelination of peripheral nerves
(D) Destruction of upper and lower motor neurons
(E) Dilation of the central spinal canal
Return the answer option letter in parenthesis () without any explanation.
Answer:

B.2.4 SciQ

Please answer with the correct option letter.
Question: Nearly all protists exist in some type of aquatic environment, including freshwater and marine environments, damp soil, and even snow. Several protist species are parasites that infect animals or plants. A few protist species live on dead organisms or their wastes, and contribute to what?
Choices:
(A) greenhouse gas
(B) habitat loss
(C) spontaneous mutation
(D) their decay
Return only the correct option letter in parenthesis () without any explanation.
Answer:

B.2.5 Social iQA

Please answer with the correct option letter based on the given context.
Context: Skylar met their requirements and ended up getting hired for the job.
Question: How would you describe Skylar?
Choices:
(A) excited to start
(B) as someone that got the job
(C) proud of herself
Return only the correct option letter in parenthesis () without any explanation.
Answer:

B.2.6 SVAMP

Solve the following math problem and provide a numerical answer.
Question: Jack received 6 emails and 8 letters in the morning. He then received 2 emails and 7 letters in the afternoon. How many more letters did Jack receive in the morning than in the afternoon?.
Give only the correct answer without explanation.
Answer:

B.2.7 Winogrande

Choose the correct option letter that fills in the underscore blank in the question.
Question: The tailor called Megan to say her coat was finished, so she asked her assistant Patricia to get it. _ drove the car to the tailor shop.
Choices:
(A) Megan
(B) Patricia
Return only the letter corresponding to the correct option in parenthesis () without any explanation.
Answer:

B.2.8 Commonsense QA

Please answer with the correct option letter.
Question: John looked for shade but couldn't find any. Where might he be?
Choices:
(A) sunny place
(B) summer
(C) direct sunlight
(D) bright sunshine
(E) full sunlight
Return only the correct option letter in parenthesis () without any explanation.
Answer:

B.2.9 LogiQA

Please answer with the correct option letter based on the given context.
Context: A and B walk from the library to the classroom at the same time. A walks halfway and runs halfway. B walks halfway and runs halfway. If both walk, the speed is the same.
Question: then?
Choices:
(A) A arrives in the classroom first.
(B) B arrives in the classroom first.
(C) A and B arrive at the classroom at the same time.
(D) Unable to judge.
Return only the correct option letter in parenthesis () without any explanation.
Answer:

B.2.10 Big Bench Hard (Boolean Expressions)

Choose the correct option letter corresponding to the boolean value of the question.
Question: not (True) and (True) is
Choices:
(A) True
(B) False
Return only the correct option letter in parenthesis () without any explanation.
Answer:

B.3 Taskwise Query Composition and Structure

Refer Table 7 for details.

Table 7: Dataset attributes used in our evaluation. Bracketed `answer_type` values denotes space of correct answers (number of options, or symbolically ∞ for numerical responses).

Attribute	SVAMP	GSM8k	HSwag	SciQ	SoQA	WinoG	MedQ	CSQA	LogiQ	BBH
num_queries	700	2000	2000	2000	2000	1000	1000	2000	2000	125
answer type	num(∞)	num(∞)	mcq(4)	mcq(4)	mcq(3)	mcq(2)	mcq(5)	mcq(5)	mcq(4)	mcq(2)

C ProxRouter Computational Overhead Analysis

We measure the delay overhead for routing a single query through ProxRouter across three different scenarios. We analyze *(a)* the delay for nearest neighbor based scenario with varying number of training queries, *(b)* the delay for nearest neighbor based routing across different choices of k, *(c)* delay for clustering based router for varying number of clusters; all for multiple choice for encoding dimension $d_{\text{enc}} \in \{128, 256, 384, 512, 768\}$. kNN-Prox computational graph includes fetching neighbors–calculating tilted aggregation weights–estimating model utilities–calculating argmax over models. KM-Prox computational graph involves calculating the cluster weights–computing the estimates of model objective and finally taking an argmax.

We utilize FAISS Douze et al. (2025) and Scikit-Learn Pedregosa et al. (2011) for index search over query points in the encoding space. For stable comparison across different parameters values, we pin all math libraries to one CPU thread which avoids multithreading jitters and dynamic thread allocation delays.

We highlight that the time delay overhead is in the millisecond range, and thus two orders of magnitude smaller than the usual generation latency of LLMs due to their autoregressive nature. Additionally, we only need to store the training query encodings and model accuracy-cost values, which has a memory footprint of 35MB for float32 representation of 768-dimensional encodings for $10,000$ training queries. ProxRouter computational graph can easily fit on a single CPU thread, further underscoring its efficiency. Hence, ProxRouter practically adds negligible computational and time overhead to query routing.

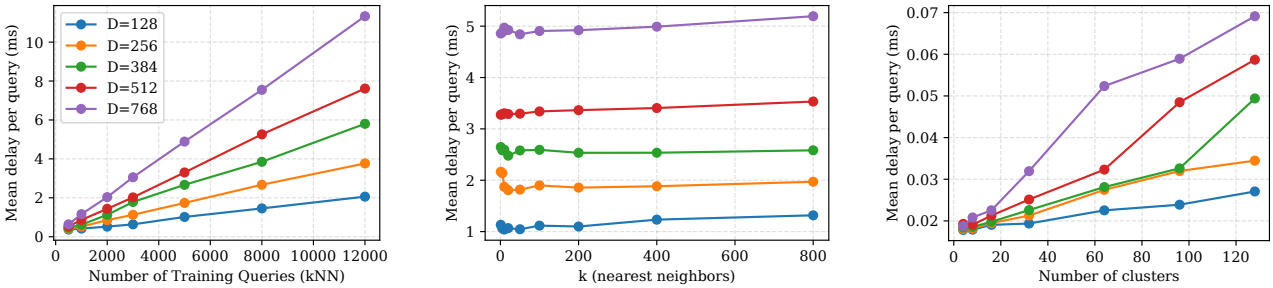


Figure 9: Routing overhead for ProxRouter (in milliseconds). *Left* plot describes delay of kNN-Prox for varying number of training queries $|\mathcal{X}_{\text{tr}}|$, *center* plot describes kNN-Prox latency for varying number of nearest neighbors k, *right* plot depicts KM-Prox latency for different number of clusters. In all cases, the overhead is roughly 100x smaller than the LLM generation latency, which is in the order of seconds due to autoregressive nature of generation Kwon et al. (2023).

D Minimum Variance Priors

For a general set of samples $\{x_i = (\mu + \varepsilon_i) : i \in [n]\}$, with zero mean noise $\text{Var}[\varepsilon_i] = \sigma_i^2$, any unbiased estimate or weighted average from the family

$$\widehat{\mu} = \sum_{i \in [n]} w_i \, x_i \, , \qquad w_i \geq 0, \;\; \sum_i w_i = 1.$$

has variance $\sum_i w_i^2 \, \sigma_i^2$. For the uniform sample variance case ($\sigma_i^2 = \sigma_j^2 \;\; \forall \, i, j \in [n]$), the variance of the estimate $\widehat{\mu}$ is

$$\text{Var}(\widehat{\mu}) = \sum w_i^2 \sigma_i^2$$

Any deviation from uniform averaging weights over the same $[n]$ points yields a strictly higher variance estimate for $\widehat{\mu}$.

For heteroscedastic noise case, optimal least variance estimate weights $w_i \propto 1/\sigma_i^2$, and the minimum variance estimator is

$$\widehat{\mu} = \sum_{i \in [n]} w_i x_i = \sum_i \frac{1/\sigma_i^2}{\sum 1/\sigma_i^2} x_i$$

$\text{Var}(\widehat{\mu}) = 1/\sum_i \sigma_i^{-2}$. Any deviation from these aggregation weights in the heteroscedastic setting yields a strictly higher variance estimate. The above result can be shown by using the Lagrange multipliers method for minimizing total variance $\sum_i w_i^2 \sigma_i^2$ subject to unbiasedness constraint $\sum_i w_i = 1$, and finding the stationary point for

$$\mathcal{L}(\mathbf{w}, \lambda) = \sum_i w_i^2 \sigma_i^2 + \lambda \big(\sum_i w_i - 1 \big)$$

which results in

$$\frac{\partial \mathcal{L}}{\partial w_i} = 2 w_i \sigma_i^2 + \lambda = 0 \quad \Rightarrow \quad w_i = -\frac{\lambda}{2\sigma_i^2}$$

And normalizing weights provides $w_i = \sigma_i^{-2} / \sum_i \sigma_i^{-2}$.

Thus, our proximity-weighted aggregation estimator suffers from the unavoidable increase in variance in exchange for a smaller bias in the estimate. Crucially, we provide a tunable knob through the τ parameter that controls such bias-variance tradeoff and tames the squared error in objective value estimation.

E Proof of eq. (8)

To Prove:

$$\min_{\mathbf{w}(\cdot)\in\Delta^{|\mathcal{I}|}} \sum_{i\in\{1,\cdots,|\mathcal{I}|\}} w_i(\mathbf{x})\,\phi_i(\mathbf{x}) + \tau\, D_{\mathrm{KL}}(\mathbf{w}(\mathbf{x})\|\mathbf{p}(\mathbf{x}))$$
$$\Rightarrow w_i(\mathbf{x}) \propto p_i(\mathbf{x})\, e^{-\phi_i(\mathbf{x})/\tau}.$$

Proof: The objective is strictly convex on $\Delta^{|\mathcal{I}|}$, so the minimizer is unique. Form the Lagrangian with the equality constraint $\sum_i w_i = 1$:

$$\mathcal{L}(\mathbf{w},\lambda) = \sum_i w_i\phi_i + \tau\sum_i w_i \log\frac{w_i}{p_i} + \lambda\left(\sum_i w_i - 1\right).$$

Stationarity ($\partial\mathcal{L}/\partial w_i = 0$) gives, for any i with $w_i > 0$,

$$\phi_i + \tau\Big(1 + \log\tfrac{w_i}{p_i}\Big) + \lambda = 0 \quad\Longrightarrow\quad w_i \;=\; C\,p_i\,e^{-\phi_i/\tau},\;\; C := e^{-(\lambda+\tau)/\tau}.$$

Enforcing $\sum_i w_i = 1$ yields $C = \big(\sum_j p_j e^{-\phi_j/\tau}\big)^{-1}$ and hence

$$w_i^* = \frac{p_i\, e^{-\phi_i/\tau}}{\sum_j p_j\, e^{-\phi_j/\tau}}.$$

□

F Bias Variance Decomposition

Consider the objective function in eq (1) to be calculated/estimated for any query $\mathbf{x}\in\mathcal{X}$, model m

$$U^{(m)}(\mathbf{x}) = \mathrm{acc}^{(m)}(\mathbf{x}) - \lambda\,\mathrm{cost}^{(m)}(\mathbf{x})$$

As the objective value is a random quantity due to the inherent uncertainty in generation from language models, leading to variable response correctness and output lengths (and consequently costs), we decompose the objective into

$$U^{(m)}(\mathbf{x}) = \mathbb{E}[U^{(m)}(\mathbf{x})] + \epsilon^{(m)}(\mathbf{x})$$

where $\overline{U}^{(m)}(\mathbf{x}) = \mathbb{E}[U^{(m)}(\mathbf{x})]$ is the conditional expectation of the objective for model m given query $\mathbf{x}$, and $\epsilon^{(m)}(\mathbf{x})$ is the zero mean noise component of each objective value associated to the randomness in generation.

F.1 Assumptions

Assumption 1. *(Local Smoothness Under Dissimilarity Measure) For any given queries $\mathbf{x}_1$ and $\mathbf{x}_2$, the true objective value difference is upper bounded by a global constant times the distance/dissimilarity between the queries.*

$$|U^{(m)}(\mathbf{x}_1) - U^{(m)}(\mathbf{x}_2)| \le L\; d(\mathbf{x}_1,\mathbf{x}_2) \quad \forall\,\mathbf{x}_1,\mathbf{x}_2\in\mathcal{X};\;\; m\in\mathcal{M} \tag{9}$$

Where $d(\cdot,\cdot)$ is any measure that captures dissimilarity between queries in the encoding space. In our analysis, we concern ourselves with dissimilarity measures in the encoding space based on the training objectives of the encoder models (such as cosine distance, Euclidean distance etc.).

Assumption 2. *(Centroids as unbiased Cluster Summaries) For any cluster c_i for $i\in[|\mathcal{I}|]$ of query embeddings, the average of objective functions at query points in a cluster acts as an unbiased estimator of the objective value at the centroid.*

$$\mathbb{E}\Big[\frac{1}{n_i}\sum_{\mathbf{x}_{tr}\in c_i} U^{(m)}(\mathbf{x}_{tr})\Big] \triangleq \mathbb{E}\left[V_i^{(m)}\right] = \overline{U}^{(m)}(\mathbf{r}_i)$$

where $\mathbf{r}_i$ is the cluster centroid for cluster c_i and $V_i^{(m)}$ is the average objective value for queries in a cluster as per the reference set notation.

Assumption 3. *(Independence of Evaluation Noise $\epsilon^{(m)}(\mathbf{x})$) The noise associated with the objective value is independent for all queries, training and test alike.*

Assumption 4. *(Fixed Design Setting) Router is built using a fixed corpus of training queries, with deterministic encodings and observed correctness values and incurred costs.*

F.2 General Form of Non-Parametric Routers

Utilizing the reference set notation, we denote the family of nonparametric routers as estimators of model objective values for any test query $\mathbf{x}$ as

$$\widehat{U}^{(m)}(\mathbf{x}) = \sum_{i \in |\mathcal{I}|} w_i(\mathbf{x}) V_i^{(m)} \tag{10}$$

where $w_i(\mathbf{x}) \geq 0$, $\sum_{i \in |\mathcal{I}|} w_i(\mathbf{x}) = 1$. Details elaborated in section 2.2.

F.3 Explicit Bias Variance Tradeoff

Objective: Find a nonparametric estimator $\widehat{U}^{(m)}(\mathbf{x})$ from the above family of estimators that approximates the expected objective value $\overline{U}^{(m)}(\mathbf{x})$ for any test query $\mathbf{x} \in \mathcal{X}$, given the observed $U^{(m)}(\mathbf{x}_{\text{tr}})$ values for training queries $\mathbf{x}_{\text{tr}} \in \mathcal{X}_{\text{tr}}$ for all models $m \in \mathcal{M}$. The final routing decision is based on these estimates as $m^*(\mathbf{x}) \leftarrow \arg\max_m \widehat{U}^{(m)}(\mathbf{x})$.

Squared Error Decomposition: We represent the mean squared error loss between the estimate $\widehat{U}^{(m)}(\mathbf{x})$ and actual value $U^{(m)}(\mathbf{x})$ for a given test query $\mathbf{x} \in \mathcal{X}$ as

$$\begin{aligned} \text{Error}(\widehat{U}^{(m)}(\mathbf{x}), U^{(m)}(\mathbf{x})) &= \mathbb{E}\big[\big(\widehat{U}^{(m)}(\mathbf{x}) - U^{(m)}(\mathbf{x})\big)^2\big] \\ &= \mathbb{E}\big[\big(\widehat{U}^{(m)}(\mathbf{x}) - (\overline{U}^{(m)}(\mathbf{x}) + \epsilon^{(m)}(\mathbf{x}))\big)^2\big] \\ &= \mathbb{E}\big[\big(\widehat{U}^{(m)}(\mathbf{x}) - \overline{U}^{(m)}(\mathbf{x})\big)^2\big] + \mathbb{E}[\big(\epsilon^{(m)}(\mathbf{x})\big)^2] \end{aligned} \tag{11}$$

where the expectation is over the randomness in generation over the text query $\mathbf{x}$ and observations of model accuracy and cost on training queries $\mathbf{x}_{\text{tr}} \in \mathcal{X}_{\text{tr}}$. The second term is the variance of the per query zero mean noise $\epsilon^{(m)}(\mathbf{x})$ associated with the actual objective value $U^{(m)}(\mathbf{x})$. For the first term,

$$\begin{aligned} \mathbb{E}\big[(\overline{U}^{(m)}(\mathbf{x}) - \widehat{U}^{(m)}(\mathbf{x}))^2\big] &= \mathbb{E}[(\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x})) + \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x})) - \widehat{U}^{(m)}(\mathbf{x}))^2] \\ &= \mathbb{E}\big[\big(\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big)^2\big] + \mathbb{E}\big[\big(\mathbb{E}(\widehat{U}^{(m)}(\mathbf{x})) - \widehat{U}^{(m)}(\mathbf{x})\big)^2\big] \\ &\quad + 2\,\mathbb{E}\big[\big(\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big) \cdot \big(\mathbb{E}(\widehat{U}^{(m)}(\mathbf{x})) - \widehat{U}^{(m)}(\mathbf{x})\big)\big] \end{aligned}$$

After expanding and rearranging, the above expression reduces to

$$\begin{aligned} \mathbb{E}\big[\overline{U}^{(m)}(\mathbf{x}) - \widehat{U}^{(m)}(\mathbf{x}))^2\big] &= \big[\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big]^2 + \mathbb{E}\big[\big(\widehat{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big)^2\big] \\ &= \text{Bias}^2(\widehat{U}^{(m)}(\mathbf{x})) + \text{Var}(\widehat{U}^{(m)}(\mathbf{x})) \end{aligned} \tag{12}$$

Using the general representation for nonparametric routers $\widehat{U}^{(m)}(\mathbf{x}) = \sum_i w_i(\mathbf{x}) V_i^{(m)}$,

$$\begin{aligned} \text{Bias}^2(\widehat{U}^{(m)}(\mathbf{x})) &= \big[\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big]^2 = \big[\overline{U}^{(m)}(\mathbf{x}) - \mathbb{E}\Big(\sum_i w_i(\mathbf{x})\; V_i^{(m)}\Big)\big]^2 \\ &= \big[U^{(m)}(\mathbf{x}) - \sum_i w_i(\mathbf{x})\; \mathbb{E}[V_i^{(m)}]\big]^2 \end{aligned}$$

Using assumption 2, alongwith Jensen's inequality and local smoothness through assumption 1

$$\begin{aligned}\text{Bias}^2(\widehat{U}^{(m)}(\mathbf{x})) &= \Big[\sum_i w_i(\mathbf{x})\big(\overline{U}^{(m)}(\mathbf{x}) - \overline{U}^{(m)}(\mathbf{r}_i)\big)\Big]^2 \\ &\le \sum_i w_i(\mathbf{x})\big(\overline{U}^{(m)}(\mathbf{x}) - \overline{U}^{(m)}(\mathbf{r}_i)\big)^2 \\ &\le \sum_i w_i(\mathbf{x})\ L^2\ d^2(\mathbf{x},\mathbf{r}_i)\end{aligned}$$

Similarly for the variance term $\text{Var}[\widehat{U}^{(m)}(\mathbf{x})]$,

$$\begin{aligned}\text{Var}(\widehat{U}^{(m)}(\mathbf{x})) &= \mathbb{E}\big[\big(\widehat{U}^{(m)}(\mathbf{x}) - \mathbb{E}(\widehat{U}^{(m)}(\mathbf{x}))\big)^2\big] = \mathbb{E}\Big[\sum_i w_i(\mathbf{x})\big(V_i^{(m)} - \mathbb{E}[V_i^{(m)}]\big)\Big]^2 \\ &= \sum_i w_i^2(\mathbf{x})\ \text{Var}[V_i^{(m)}]\end{aligned}$$

For clustering based algorithms, $\text{Var}[V_i^{(m)}] = \mathbb{E}\Big[\frac{1}{n_i^2}\sum_{\mathbf{x}_{\text{tr}}\in c_i}(\epsilon^{(m)}(\mathbf{x}_{\text{tr}}))^2\Big] = \frac{1}{n_i^2}\sum_{\mathbf{x}_{\text{tr}}\in c_i}\text{Var}[\epsilon^{(m)}(\mathbf{x}_{\text{tr}})]$ for cluster c_i with n_i number of training queries. For nearest neighbor based routers, $\text{Var}[V_i^{(m)}] = \text{Var}[\epsilon^{(m)}(\mathbf{x}_{\text{tr}})]$ where $\mathbf{x}_{\text{tr}}$ is the training query associated to reference element $i \in |\mathcal{I}|$.

Rewriting the squared error eq. (11), and the bias and variance eq. (12) for the model objective value estimator at any test query $\mathbf{x}$, we have

$$\begin{aligned}\text{Error}(\widehat{U}^{(m)}(\mathbf{x}), U^{(m)}(\mathbf{x})) &= \text{Bias}^2(\widehat{U}^{(m)}(\mathbf{x})) + \text{Var}(\widehat{U}^{(m)}(\mathbf{x})) + \text{Var}[\epsilon^{(m)}(\mathbf{x})] \\ &\le \sum_i w_i(\mathbf{x})\ L^2\ d^2(\mathbf{x},\mathbf{r}_i) + \sum_i w_i^2(\mathbf{x})\ \text{Var}[V_i^{(m)}] + \text{Var}[\epsilon^{(m)}(\mathbf{x})]\end{aligned}$$

Remarks: For choice of $\mathbf{w}(\mathbf{x}) = [w_1(\mathbf{x}), \cdots, w_{|\mathcal{I}|}(\mathbf{x})]$ as the initial least variance prior $\mathbf{p}(\mathbf{x}) = [p_1(\mathbf{x}), \cdots, p_{|\mathcal{I}|}(\mathbf{x})]$ in section 3.1, the variance term $\sum_i p_i^2(\mathbf{x})\ \text{Var}[V_i^{(m)}]$ is minimized and any other $\mathbf{w}(\mathbf{x}) \neq \mathbf{p}(\mathbf{x})$ strictly increases variance. To reduce to bias term $\sum_i w_i(\mathbf{x})\ L^2\ d^2(\mathbf{x},\mathbf{r}_i)$, we selectively prioritize closer reference elements to the test query $\mathbf{x}$ by proximity based tilting. Choosing $w_i(\mathbf{x}) \propto p_i(\mathbf{x})\exp(-\phi_i(\mathbf{x})/\tau)$ where proximity penalty $\phi_i(\mathbf{x})$ is an increasing function of the distance $d(\mathbf{x},\mathbf{r}_i)$ between test query $\mathbf{x}$ and reference element i. Any super-logarithmic choice of proximity penalty leads to a decaying bias term in $d(\mathbf{x},\mathbf{r}_i)$, eventually leading to a balanced tradeoff for the squared error term eq. (11). Note that the last term $\text{Var}[\epsilon^{(m)}(\mathbf{x})]$ captures the unavoidable noise in generated response for query $\mathbf{x}$ by model m.

G Effective Sample Size

For any unbiased aggregation scheme over observed samples, the effective sample size corresponds to size of an unweighted sample average that would yield the variance reduction obtained through a weighted aggregation. For a general set of samples $\{x_i = (\mu + \varepsilon_i) : i \in [n]\}$, with zero mean uniform variance noise $\mathbb{E}[\varepsilon_i^2] = \sigma_i^2$, any unbiased estimate of the form

$$\widehat{\mu} = \sum_{i\in[n]} w_i\ x_i\ , \qquad w_i \ge 0,\ \ \sum_i w_i = 1.$$

has variance $\sum_i w_i^2\ \sigma_i^2$. The effective sample size for such an estimator is:

$$\text{ESS}(\widehat{\mu}) = \frac{\sum w_i^2}{(\sum w_i)^2} = \sum w_i^2$$

So $\text{Var}(\widehat{\mu}) = \sigma^2/\text{ESS}(\widehat{\mu})$. Any deviation from uniform averaging weights over the same $[n]$ points yields a strictly higher variance estimate.

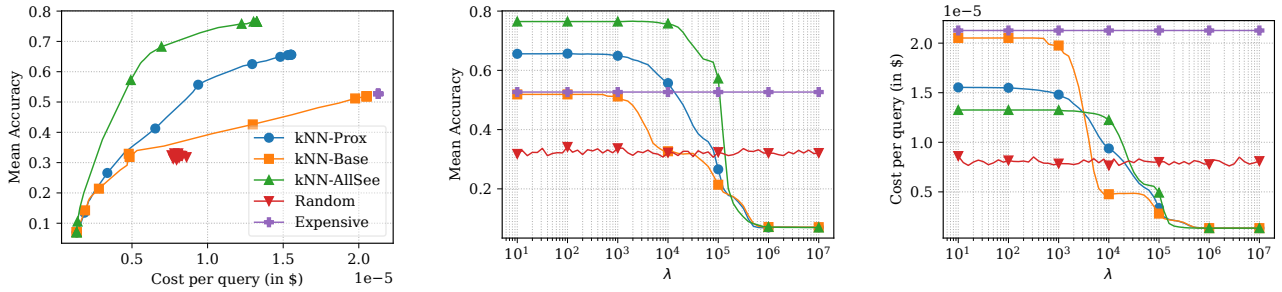
As found in Appendix D, the optimal estimator in heteroscedastic noise is $w_i \propto 1/\sigma_i^2$, and the variance of the corresponding estimator is $\mathrm{Var}(\widehat{\mu}) = 1/\sum_i \sigma_i^{-2}$. Any deviation from these aggregation weights in the heteroscedastic setting yields a strictly higher variance estimate. For our analysis of techniques in kNN based aggregation (Table 3), we utilize the closest $k = 100$ nearest neighbors of any test query to ensure fair comparison which inevitably leads to a higher variance in kNN-Prox causing a slight routing performance drop on inlier queries. Practically, kNN-Prox leads to a smaller effective sample size over the closest k queries, which can be overcome by choosing a larger set of neighboring queries.

H ProxRouter Extended Results

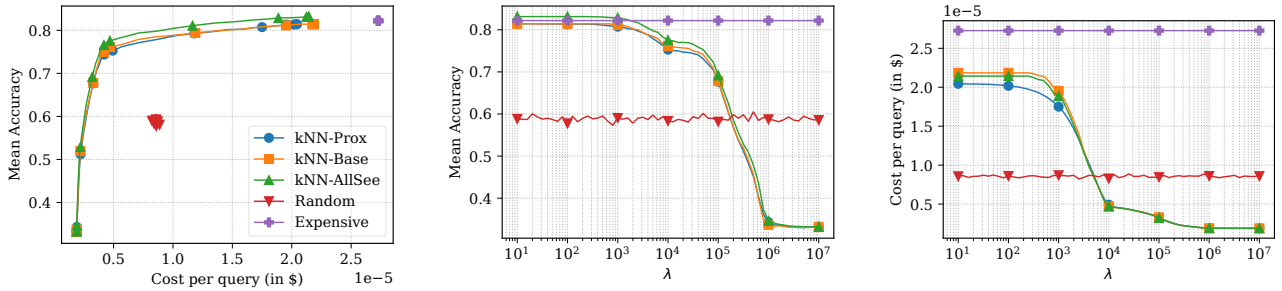
H.1 Detailed Results for Section 4

In this section, we present details on results already presented in the main segment of Section 4. Figure 10 denotes the performance of kNN-Prox as presented in Table 3.

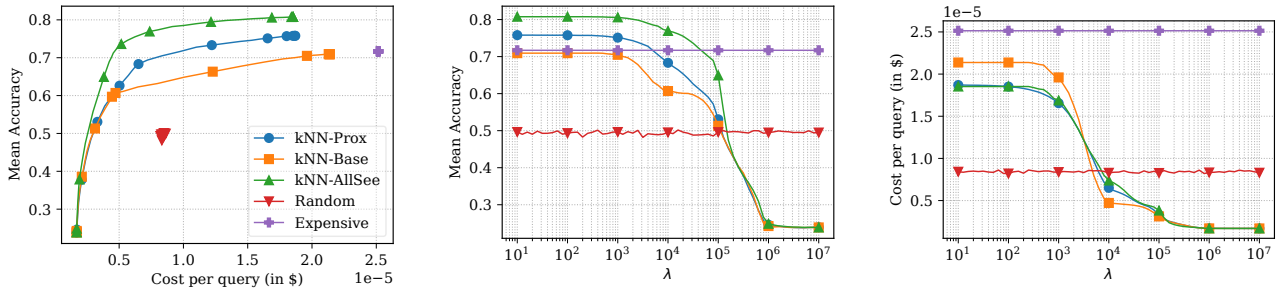
Figures 11 and 12 denote the performance of KM-Prox as presented in Table 2.



(a) Routing Performance on outlier test queries.

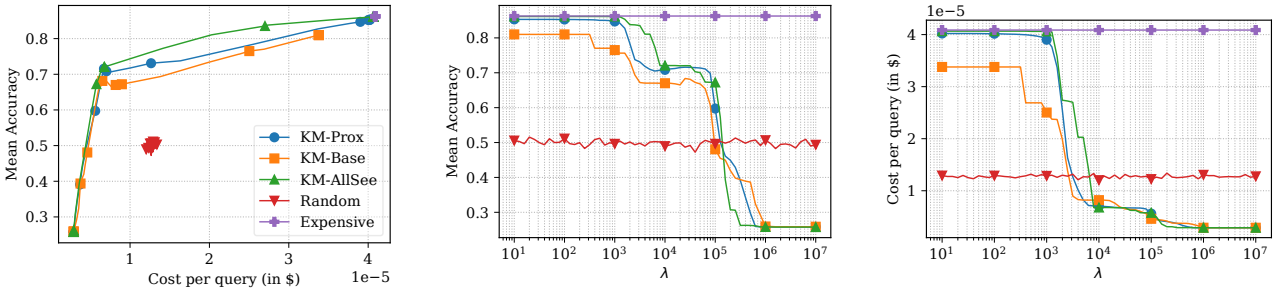


(b) Routing Performance on inlier test queries.

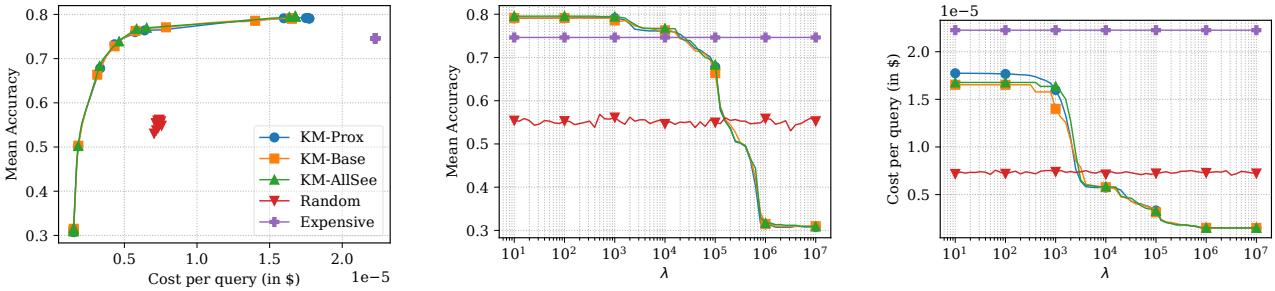


(c) Routing Performance on all test queries.

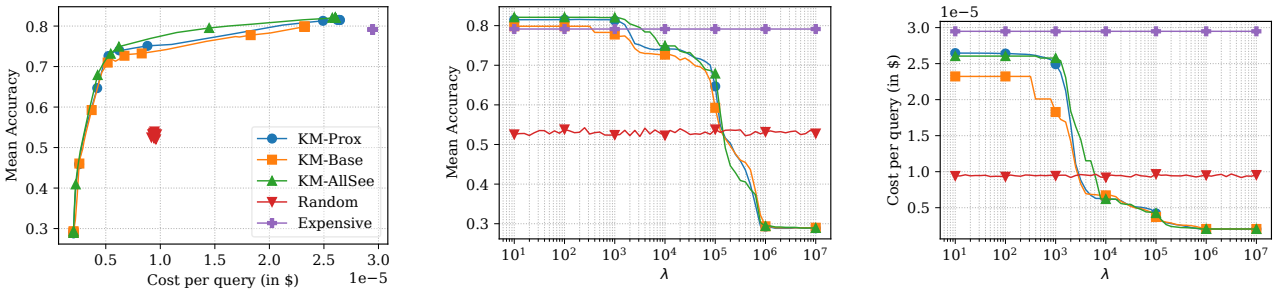
Figure 10: kNN-Prox performance on GSM8K+SVAMP outlier tasks, Table 3. Note that outlier performance is greatly improved while preserving inlier query routing performance.



(a) Routing Performance on outlier test queries.

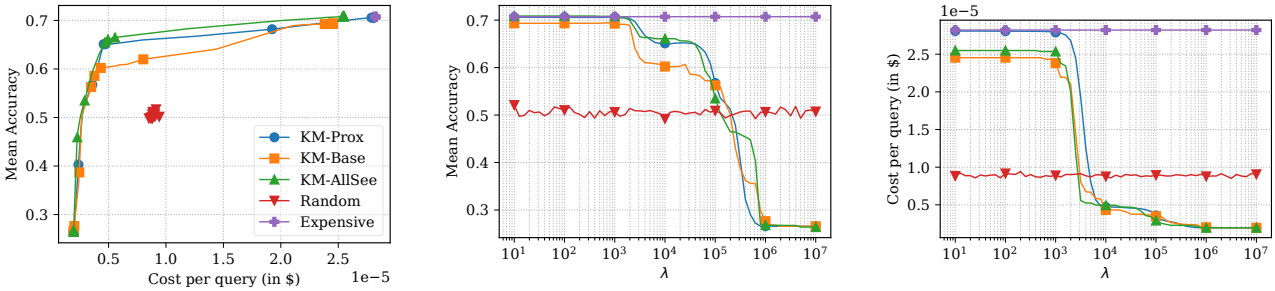


(b) Routing Performance on inlier test queries.

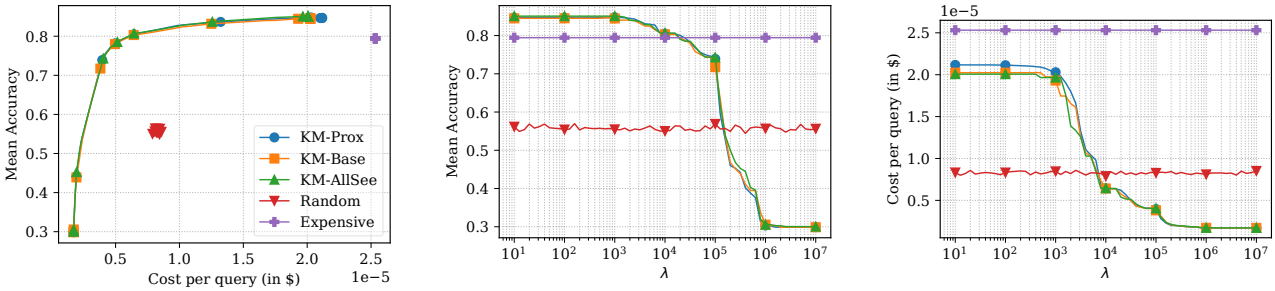


(c) Routing Performance on all test queries.

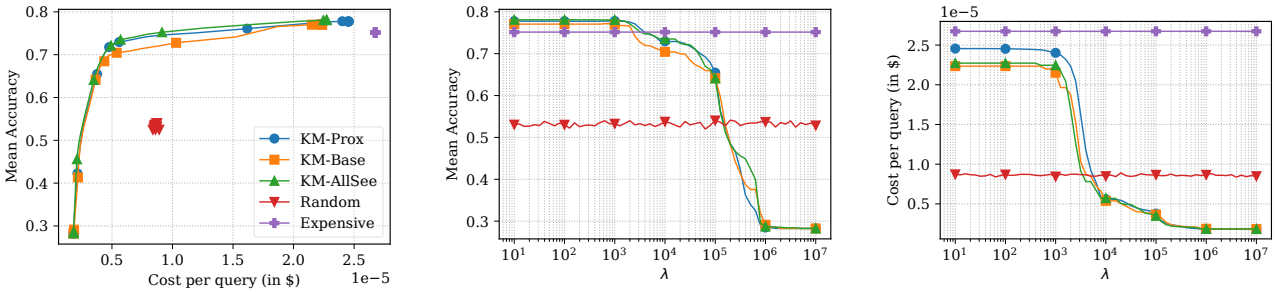
Figure 11: *K*M-Prox performance on MedQA+Hellaswag outlier tasks, Table 2. Note that outlier performance is greatly improved while preserving inlier query routing performance.



(a) Routing Performance on outlier test queries.



(b) Routing Performance on inlier test queries.



(c) Routing Performance on all test queries.

Figure 12: KM-Prox performance on LogiQA+CommonSenseQA+BBH(BoolEx) outlier tasks, Table 2. Note that outlier performance is greatly improved while preserving inlier query routing performance.

H.2 Ablations for Num clusters (KMeans Router), Num Neighbors (kNN Router)

In this section, we present the routing performance values for different number of clusters K for clustering based routers (Table 8), and for different values of nearest neighbors k for nearest neighbor based routers (Table 9).

Table 8: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for multiple values of number of clusters K. Upper bound denoted by the full knowledge router KM-AllSee.

K	Outlier Tasks	Split	Routing method		Upper Bound
			KM-Base	KM-Prox	KM-AllSee
16	Hellaswag, MedQA	Outlier	71.24%	75.11%	78.04%
		Inlier	74.77%	74.82%	75.16%
		Overall	73.64%	75.24%	76.59%
	LogiQA, CSQA, BBH	Outlier	65.60%	66.22%	67.07%
		Inlier	79.97%	80.05%	79.90%
		Overall	73.07%	73.54%	73.96%
32	Hellaswag, MedQA	Outlier	70.68%	74.88%	78.36%
		Inlier	74.62%	74.86%	74.63%
		Overall	73.04%	75.12%	74.87%
	LogiQA, CSQA, BBH	Outlier	63.39%	66.18%	67.25%
		Inlier	79.35%	79.92%	79.94%
		Overall	71.61%	73.46%	73.88%
64	Hellaswag, MedQA	Outlier	70.65%	74.72%	78.56%
		Inlier	74.27%	74.70%	75.38%
		Overall	72.96%	74.96%	76.87%
	LogiQA, CSQA, BBH	Outlier	64.02%	66.12%	67.74%
		Inlier	78.71%	79.66%	80.15%
		Overall	71.70%	73.32%	74.24%

Table 9: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for multiple values of nearest neighbors k. Upper bound denoted by the full knowledge router kNN-AllSee.

k	Outlier Tasks	Split	Routing method		Upper Bound
			kNN-Base	kNN-Prox	kNN-AllSee
50	GSM8k, SVAMP	Outlier	44.52%	47.27%	61.60%
		Inlier	76.46%	76.23%	79.66%
		Overall	66.00%	68.28%	75.00%
100	GSM8k, SVAMP	Outlier	38.55%	46.64%	60.77%
		Inlier	77.51%	76.96%	79.11%
		Overall	63.98%	68.12%	74.60%
200	GSM8k, SVAMP	Outlier	38.83%	46.10%	59.60%
		Inlier	77.83%	77.08%	78.85%
		Overall	64.15%	67.61%	74.35%

H.3 Ablation on Distance Metric Choice

Tables 10 and 11 denote the performance of our approach with Euclidean Distance as the choice of distance between points in the query encoding space. KM-Prox and kNN-Prox outperform KM-Base and kNN-Base across both Euclidean Distance and Cosine Distance as metrics, highlighting the distance-metric agnostic improvement yielded by proximity weighted aggregation which controls the bias-variance tradeoff.

Table 10: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for chosen sets of outlier tasks. Distance Metric: Euclidean

Outlier Tasks	Split	Routing method		Upper Bound
		KM-Base	KM-Prox	KM-AllSee
Hellaswag, MedQA	Outlier	74.23%	76.04%	78.62%
	Inlier	74.64%	75.00%	74.92%
	Overall	74.51%	75.68%	76.68%
LogiQA, CSQA, BBH	Outlier	64.03%	66.49%	67.39%
	Inlier	79.45%	80.06%	79.82%
	Overall	71.94%	73.73%	73.98%

Table 11: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for chosen outlier tasks. Distance Metric: Euclidean Distance.

Outlier Tasks	Split	Routing method		Upper Bound
		kNN-Base	kNN-Prox	kNN-AllSee
GSM8k, SVAMP	Outlier	38.58%	46.68%	60.66%
	Inlier	77.51%	76.89%	79.11%
	Overall	63.97%	67.97%	74.60%

H.4 Ablation on Choice of Sentence Encoder

Tables 12 to 14 denote the performance of ProxRouter on clustering based routers which different choices of sentence encoders: `paraphrase_albert_small_v2_embedding` Reimers and Gurevych (2019), `all_minilm_l6_v2_embedding` Wang et al. (2020), `all_distilroberta_v1_embedding` Sanh et al. (2019). Across all choices of sentence encoders, our approach consistently improves routing performance as compared to the baseline.

Table 12: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for chosen sets of outlier tasks. Sentence Encoder: `paraphrase_albert_small_v2_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		KM-Base	KM-Prox	KM-AllSee
Hellaswag, MedQA	Outlier	72.09%	76.75%	78.23%
	Inlier	75.14%	75.06%	76.10%
	Overall	73.99%	75.84%	77.14%
LogiQA, CSQA, BBH	Outlier	64.87%	66.40%	67.73%
	Inlier	80.48%	80.21%	80.67%
	Overall	72.79%	73.63%	74.58%

Table 13: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for chosen sets of outlier tasks. Sentence Encoder: `all_minilm_l6_v2_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		KM-Base	KM-Prox	KM-AllSee
Hellaswag, MedQA	Outlier	72.00%	75.09%	78.40%
	Inlier	73.78%	74.46%	74.90%
	Overall	73.19%	74.90%	76.51%
LogiQA, CSQA, BBH	Outlier	64.02%	66.36%	67.13%
	Inlier	79.49%	80.03%	79.96%
	Overall	71.97%	73.53%	73.83%

Table 14: Performance (AUC normalized) of KM-Prox (ProxRouter) vs. KM-Base for chosen sets of outlier tasks. Sentence Encoder: `all_distilroberta_v1_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		KM-Base	KM-Prox	KM-AllSee
Hellaswag, MedQA	Outlier	74.96%	77.18%	78.39%
	Inlier	75.17%	74.94%	76.31%
	Overall	74.99%	75.76%	77.32%
LogiQA, CSQA, BBH	Outlier	65.31%	66.48%	67.88%
	Inlier	80.74%	80.29%	81.08%
	Overall	73.39%	73.73%	74.87%

Tables 15 to 17 denote the performance of ProxRouter on nearest neighbor aggregation based routers which different choices of sentence encoders: `paraphrase_albert_small_v2_embedding` Reimers and Gurevych (2019), `all_minilm_l6_v2_embedding` Wang et al. (2020), `all_distilroberta_v1_embedding` Sanh et al. (2019). Across all choices of sentence encoders, our approach consistently improves routing performance as compared to the baseline.

Table 15: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for chosen outlier tasks. Sentence Encoder: `paraphrase_albert_small_v2_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		kNN-Base	kNN-Prox	kNN-AllSee
GSM8k, SVAMP	Outlier	39.72%	51.27%	61.42%
	Inlier	77.45%	76.43%	78.87%
	Overall	64.79%	69.86%	74.70%

H.5 Parametric and Nonparametric Router Comparison

We compare nonparametric approaches with a parametric approach utilizing learned neural network router (MLP layers) on top of the query sentence encodings. Our MLP router utilizes a two head output where each head's output dimension is the number of models in the pool ($|\mathcal{M}|$), one head predicting accuracy and the other head predicting cost of response for all models. Model selection is done through the same approach, where eq. (1) is now calculated using these accuracy and cost estimates for each model for a given value of λ. We also reiterate that MLP router needs full retraining to accommodate a change in the model pool or addition of newer tasks in the training set, unlike KMeans or kNN based routers which can naturally accommodate such changes in an incremental manner. We use the same experimental setting in Table 1, with first half tasks as outliers and other half as inliers. Figure 13 presents our comparison of parametric (MLP Router) and nonparametric (kNN and KMeans routers) query routing approaches, highlighting that nonparametric methods match or even outperform complicated parametric routers.

Table 16: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for chosen outlier tasks. Sentence Encoder: `all_minilm_l6_v2_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		kNN-Base	kNN-Prox	kNN-AllSee
GSM8k, SVAMP	Outlier	38.92%	47.99%	59.48%
	Inlier	77.16%	75.97%	78.85%
	Overall	63.98%	69.20%	74.17%

Table 17: Performance (AUC normalized) of kNN-Prox (ProxRouter) vs. kNN-Base for chosen outlier tasks. Sentence Encoder: `all_distilroberta_v1_embedding`

Outlier Tasks	Split	Routing method		Upper Bound
		kNN-Base	kNN-Prox	kNN-AllSee
GSM8k, SVAMP	Outlier	39.34%	55.14%	60.98%
	Inlier	77.49%	76.63%	79.00%
	Overall	64.50%	71.81%	74.69%

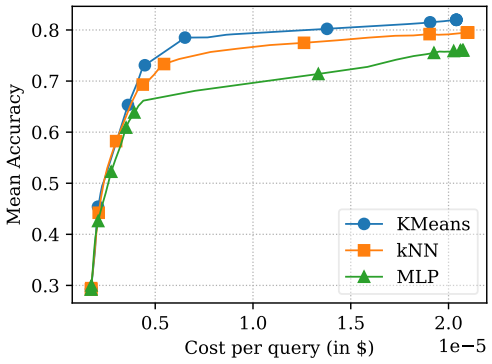


Figure 13: Nonparametric routers match, or even outperform learned parametric approaches such as MLP based routers. We utilize a two head MLP layer for predicting accuracies and costs of model responses respectively, and compare routing performance with existing kNN and KMeans approaches. Same experimental setting used from Table 1.

I Reproducibility Checklist

We generate responses from LLMs for all prompts using a fixed seed (`42`). The query encodings generated from sentence encoders are deterministic, and so are the encodings of training queries. We run KMeans clustering algorithm using a constant seed (`42`) for initialization of cluster centers, but we observed almost indistinguishable results for other seeds as well, indicating stable clustering. Nearest neighbor based schemes are fully deterministic due to the deterministic nature of encodings, hence we present the results directly. All train-test splits are also created deterministically with the same choice of random seed. Our code is provided in supplemental zip submission. For additional analysis such as delay modeling, we repeat our experiments over five different seeds $(7, 42, 99, 1234, 2024)$ and present the average results. We furthermore pin the experimental setting in delay analysis by choosing one CPU thread for avoiding multi-threading jitters and dynamic BLAS thread allocation artifacts for reflecting true time complexity of our approach.