TWO-STAGE LLM ROUTING WITH EMBEDDING REPRESENTATION

Zeng Weixuan*
School of Data Science, The Chinese University of Hong Kong, Shenzhen, China
weixuanzeng@link.cuhk.edu.cn

ABSTRACT

In this work, we propose a two-stage LLM routing method via embedding representation. LLM routing mainly focuses on how to find an optimal LLM for each query to balance the performance and cost. Previous related works mainly train a classifier to select the best models, which can not deal with the cost constraints well and can not adapt to new models. To tackle these problems, our method first clusters all LLM candidates into several LLM groups, and in inference stage, we just select the best suitable groups and find the optimal LLM from the selected groups. In addition, we optimize a projection matrix to link the performance of LLMs and their embeddings, which is convenient to add new models as we just need obtain their embeddings via the projection matrix. Experiments demonstrate that our method can achieve better performance than all single models and baseline routing methods.

1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable capabilities across various tasks. Many LLMs are publicly available online, such as Qwen2.5 (Team 2024), Deepseek-V3 (DeepSeek-AI 2024), and LLaMA-3 (Grattafiori et al. 2024). Those LLMs have been further fine-tuned to be generalists or specialists. For example, Deepseek-R1-Distill-Qwen Models (DeepSeek-AI 2025) are fine-tuned from Qwen Models using GRPO (Shao et al. 2024) technique, which excel in solving the difficult math and reasoning tasks. Besides, lots of proprietary models also own impressive abilities, like GPT, Mistral and Claude models (Radford et al. 2019 ; Brown et al. 2020; Achiam et al. 2023; Claude 2024; Team et al. 2023). All of these advanced LLM models have prompted the development of AI community and improved our efficiency. However, achieving more strong capabilities usually requires LLMs of larger parameters and substantial computational resources (Zhang et al. 2025). More precisely, their impressive abilities notwithstanding, the inference cost of LLMs can be prohibitive, which has has motivated a range of techniques to improve inference efficiency, such as speculative decoding, early-exiting, pruning, distillation, and others (Chen et al. 2023a; Schuster et al. 2022; Agarwal et al. 2024; Aishwarya et al. 2024). Most of them need retraining the LLM or adjusting the architecture of models, which takes much compute resources and time, and may not remain feasible or scalable in long term, especially when new models emerge daily (Hu et al. 2024;).

Another interesting fact is that although larger LLMs generally offer better performance, there is still some problems they can not solve or just answer incorrectly. As shown in Figure 1, on the RouterBench (Hu et al. 2024) dataset, the 7B small model `mistral-7B-chat` with 26% performance lower than `gpt-4`, exhibits some stronger ability on solving problems whose ID fall between 5000 and 10000. This demonstrates that every individual model has its own speciality and no single model can address all instructions. It is natural to think how about ensembling multiple models to boost performance. Right! Ensembling all of these models can achieve 77% accuracy, which is much higher than any single model. However, this method require querying all the LLMs at least $O(T)$ times in inference, and is still computationally prohibitive.

Based on above mentioned background, the focus of our project is about how to balance the efficiency and cost, namely assembling multiple off-the-shelf LLMs to harness their complementary

*Thanks for the instructor Wang Benyou and PHD Wang Rongsheng's valuable suggestions.

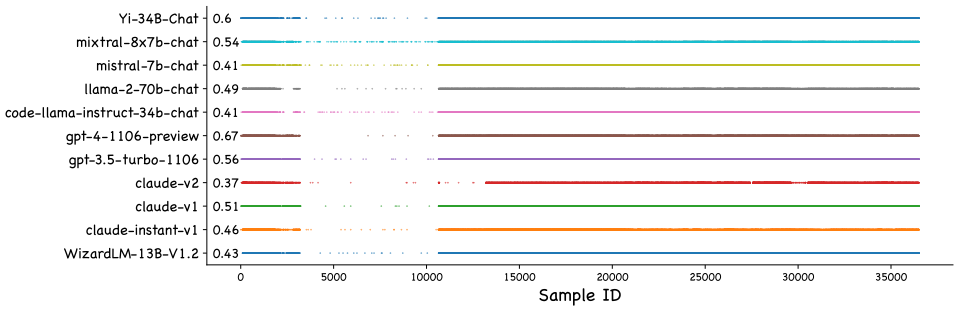

Figure 1: Answer correctness of diverse LLMs on RouterBench. The dots in the figure means that the model can solve corresponding data sample. The digits in the left are the accuracies for LLMs.

abilities and resulting in better performance with lowest cost (Chen et al. 2024), and our main method is **LLM Routing**. During inference, routing is much more efficient than vanilla ensembling because it only needs to implement inference once with the selected LLM. The key to LLM routing is to efficiently identify the optimal model from a vast pool of options, without prior access to the potential candidates' output (Zhang et al. 2025). Recent works have proposed several routing methods to select a suitable LLM for each query. Zooter (Shnitzer et al. 2023) utilizes a reward model to score all candidate LLMs' response, then trains a routing function to learn each model's expertise with the normalized rewards as supervision. Chen et al. 2024 finds that the loss of Zooter, minimizing KL divergence (Kullback & Leibler 1951) is inappropriate when multiple LLMs perform well for a query, and instead they propose RouterDC, leveraging two contrastive losses to learn the candidates' embedding representations. On learned embedding representations, we can choose the optimal LLM via computing the similarity between each query and LLMs. MODEL-AST (Zhang et al. 2025) also learn LLM's embedding representation by encoding a natural language which helps to include model's expert knowledge. However, both of RouterDC and MODEL-AST only aim to select the best-performance model, ignoring the cost constraint. Besides, the serious drawback of them is they can not generalize to new models, when new models become candidates, they need to retrain their models, which is troublesome.

To tackle these problems, we propose a Two-Stage LLM Routing with Embedding Representation. Instead of straightly learning LLM's embedding representation, our method first clusters all LLM candidates into several groups and sort all LLM candidates based on each LLM's performance and cost. For each query, we select the top-k LLM candidates as the positive models and the rest as the negative models based on their rankings. Not only LLM, we also select the LLM groups which have the most positive LLM candidates as the positive groups and the rest groups as the negative groups. Then we utilize two contrastive loss to learn the groups' and LLM's embedding representations. To adapt to the situations where new LLMs are available, we obtain each LLM candidate's performance vector by testing it on a validation dataset, then in order to align each LLM's embedding and its performance vector, we additionally train a projection matrix to make the product of the matrix and model embedding get close to the performance vector via a MSE loss. Based on these embedding, in inference stage, we can first find the most suitable LLM cluster, and then select the optimal LLM candidate from the cluster. When new model is available, we just need to test it on the validation set to obtain its performance vector, then the vector multiply the inverse of our projection matrix to obtain the model's embedding representation. Because our method does not require training any classifier, it is convenient to merge new models into the routing system.

We implement extensive experiments on two benchmark tasks: EmbedLLM (Zhuang et al. 2024) and RouterBench (Hu et al. 2024) to demonstrate the effectiveness of our methods. Empirical results show that our two-stage routing method can reach **113%** and **95%** of the performance of the best models with only **11%** and **5.6%** of cost on EmbedLLM and RouterBench, respectively. For new model situations, our method still achieves competitive performance.

In summary, our contributions are:

(i) We propose an efficient framework to construct a LLM router to select the optimal LLM for each query via two contrastive losses to learn embedding representations, which consists of LLMs' and LLM groups' representations.

(ii) To adapt to the new models, we add a MSE loss to learn a projection matrix, which is beneficial to compute new models' embedding and avoids retraining the whole system.

(iii) Experiments verify the high performance and efficiency of our routing method, which can achieve competitive performance and generalization ability using a small costs.

2 Related work

LLM Ensembling. The aim of ensembling LLM is to utilize multiple LLMs' capability to lift the overall performance. Lots of ensembling techniques have been proposed to generate improved outputs from all of the LLM candidates (Chen et al. 2024; Jitkrittum et al. 2025). Voting is a simple and effective ensemble method, which calls all the LLMs to get the most frequent response as the final output (Li et al. 2024). LLM cascading is another ensembling approaches. It query a list of LLMs sequentially until some confidence metrics reaches a prefixed threshold, then outputs the answer(Aggarwal et al. 2024; Yue et al. 2023; Chen et al. 2023b). A grievous drawback of ensembling is that it needs to query all LLM candidates, resulting in significant cost and latency increasement. Instead of querying all LLMs, routing is much more efficient because it only selects the optimal LLM to call.

LLM Routing. Many works have proposed noval and efficient framework to implement LLM routing. Zooter utilizes the external reward model to score all LLMs' response as the supervision signal, in order to train a classifier to learn the expertise of LLM candidates (Shnitzer et al. 2023). FORC (Šakota et al. 2024) formulates the query assignment problem as a integer linear programming (ILP) problem, which can allow the user to adjust the performance and cost constraints. RouteLLM (Ong et al. 2024) leverages human preference data and employs data augmentation techniques to enhance performance. GraphRouter (Feng et al. 2024) constructs a heterogeneous graph to capture the contextual information between the query' s requirements and the LLMs' capabilities. MixLLM (Wang et al. 2025) leverages query tags to enhance query embeddings and design lightweight prediction models to estimate the response qualities and costs of queries over LLMs. RouterDC (Chen et al. 2024) uses two contrastive losses to learn the candidates' embedding representations and compute cosine similarity to select the LLM without considering the cost constraints. And none of them solves the new model's adaptation problem. Our method aims to balance the efficiency and cost of querying LLMs, and avoid retraining the system to merge new models.

3 Methodology

In this section, we will show our framework of LLM routing as shown in Figure 2.

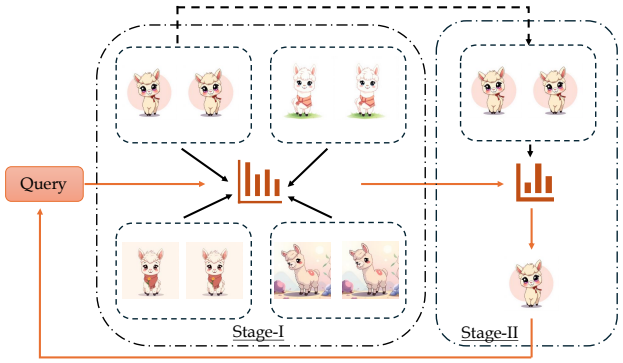

Figure 2: Our two-stage LLM routing framework.

3.1 PROBLEM DEFINITION

Given a training dataset $\mathcal{D}_{train} = \{(x_i, y_i)|i \in \{1, ...n\}\}$, x_i is a query or instruction, y_i is corresponding ground truth. Usually, we have a lots of LLM candidates $\{\mathcal{M}_t|t \in \{1, ..T\}\}$, and each LLM may generate different response. Any user inputs query x_i, then the routing system selects the suitable LLM $\mathcal{M}_t$ to answer and returns the answer to the user.

3.2 PERFORMANCE VECTOR

As mentioned before, our routing method needs the models' performance vector to adapt to new models. Before training, we split a small validation dataset $\mathcal{D}_{val} = \{(x_i, y_i)|i \in \{1, ...m\}\}$ from training dataset, which consists of $\mathcal{S}$ diverse sub-datasets $\mathcal{D}^s_{val} = \{(x_i, y_i)^s\}|i \in \{1, ...m_s\}, s \in \{1, ...S\}$. To accurately measure single LLM's characteristic, these sub-datasets will encompass a large range of domains, like math, coding, QA, reading comprehension and so on. Every LLM is tested on the validation dataset and the performance metrics on every sub-datasets constitutes the performance vector:

$$\mathcal{V}_t = (score(\mathcal{D}^1_{val})_t, ...score(\mathcal{D}^s_{val})_t, avg(score(\mathcal{D}^{1,...s}_{val})_t), std(score(\mathcal{D}^{1,...s}_{val})_t), cost(\mathcal{M}_t)), \quad (1)$$

where $\mathcal{V}_t \in \mathbb{R}^{s+3}$, $score(\mathcal{D}^s_{val})_t = \frac{1}{m_s}\sum_{m_s=1}^{m_s} \mathbf{1}(y^s_i = \hat{y^s_i})_t$ measures the performance of model $\mathcal{M}_t$ on validation sub-dataset $\mathcal{D}^s_{val}$. For classification tasks, $\mathbf{1}(y^s_i = \hat{y^s_i}) \in \{0, 1\}$, and for open-ended generation tasks, $\mathbf{1}(y^s_i = \hat{y^s_i}) \in [0, 1]$. To include more features about each LLM, we also add the cost of each LLM, and the average and standard variation of performance on these validation sub-datasets into $\mathcal{V}_t$.

3.3 LLM GROUPS

Instead of formulating the cost constraints into the optimization objectives like FORC, we cluster all LLM candidates into p groups based on LLM' performance vectors. To reduce costs, we only need to choose the lowest-cost group which can solve the query, and then find the optimal LLM from the chosen group. This group-selection method plays its role like a hard constraint, and reduces its impact on selecting the suitable LLM.

We use kmeans clustering method to assign each LLM's group:

$$\mathcal{G}_p = \{\mathcal{M}_{p1}, \mathcal{M}_{p2}, ..\mathcal{M}_{pq}|q \in \{1, ...t\}\}, \mathcal{G}_i \cap \mathcal{G}_j = \emptyset, i, j \in \{1, ..p\}, \quad (2)$$

where $\mathcal{G}_p$ consists of some LLMs and every LLM groups can have q LLMs. The q of different groups can be different, and one LLM can be included in one and only one group.

3.4 CONTRASTIVE LOSS

To select the most suitable group and LLM, for a query x_i, we encoder it into an embedding representation $\mathcal{E}_{x_i} \in \mathbb{R}^d$ using a small decoder model, d is the embedding dimension. Then we compute the probability distribution of selecting the best groups:

$$P(x_i, \mathcal{G}; \boldsymbol{\theta}) = softmax[sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_1}), sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_2}), ...sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_p})], \quad (3)$$

where $\boldsymbol{\theta}$ denotes all the trainable parameters, sim denotes the cosine similarity function, and $\mathcal{E}_{\mathcal{G}_p}$ denotes the embedding representation of LLM group $\mathcal{G}_p$. From Equation 3, we can obtain the top-k suitable groups, then we can select the optimal LLM from the top-k groups:

$$P(x_i, \mathcal{M}; \boldsymbol{\theta}) = softmax[sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{p1}}), sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{p2}}), ...sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{pq}})], \quad (4)$$

where $\mathcal{E}_{\mathcal{M}_{pq}}$ denotes the embedding representation of LLM $\mathcal{M}_{pq}$, p falls onto the top-k groups, and q is the index of the selected model from the selected groups.

In order to learn the embedding of LLMs and their groups, inspired by RouterDC, we can implement contrastive learning (Izacard et al. 2021; Ni et al. 2021). The objective of our contrastive learning is to push the embedding of one query closer to embeddings of the groups and LLMs whose answer and cost are satisfying. In more details, we sort all the LLM candidates based on their performance and costs, select the top-k LLMs as the positive models whose index set is $\mathcal{M}^+$, and the rest models as

the negative models whose index set is $\mathcal{M}^-$. In general, the positive models have better performance and lower cost than the negatives. Then we can compute the query-LLM contrastive loss:

$$\mathcal{L}_{query-LLM}(x_i, \mathcal{M}; \boldsymbol{\theta}) = \sum_{t^+ \in \mathcal{M}^+} -log \frac{e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{t^+}})}}{e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{t^+}})} + \sum_{t^- \in \mathcal{M}^-} e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{M}_{t^-}})}} \tag{5}$$

In addition, we select the LLM groups which have the most positive models as the positive groups $\mathcal{G}^+$, and the rest groups as the negative groups $\mathcal{G}^-$. So we have the query-group contrastive loss:

$$\mathcal{L}_{query-group}(x_i, \mathcal{G}; \boldsymbol{\theta}) = \sum_{t^+ \in \mathcal{G}^+} -log \frac{e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_{t^+}})}}{e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_{t^+}})} + \sum_{t^- \in \mathcal{G}^-} e^{sim(\mathcal{E}_{x_i}, \mathcal{E}_{\mathcal{G}_{t^-}})}} \tag{6}$$

3.5 PROJECTION MATRIX

When new model $\mathcal{M}_n$ is available, we first obtain its performance vector $\mathcal{V}_n$ as Section 3.2. Then we can compute its embedding representation: $\mathcal{E}_{\mathcal{M}_n} = \mathcal{C}^{-1}\mathcal{V}_n$. The aim of our projection matrix $\mathcal{C} \in \mathbb{R}^{d \times (s+3)}$ is to link the embedding representation and performance vector of the LLM. We can regard it as a hook, which makes it possible and convenient to get the desired embeddings of new models. And we include the optimization procedure into our training:

$$\mathcal{L}_{projection}(\mathcal{M}, \mathcal{V}, \mathcal{C}) = \frac{1}{T} \sum_{t=0}^{T} (\mathcal{M}_t \times \mathcal{C} - \mathcal{V}_t)^2 + norm(\mathcal{C}) \tag{7}$$

We use the MSE loss to optimize our projection matrix, and add a normalization term into the loss to avoid overfitting.

3.6 TRAINING AND INFERENCE

In training, we combine above mentioned three losses to learn our router:

$$\mathcal{L}(\mathcal{D}_{train}; \boldsymbol{\theta}) = \sum_{x_i \in \mathcal{D}_{train}} \alpha \mathcal{L}_{query-LLM}(x_i, \mathcal{M}; \boldsymbol{\theta}) + \mathcal{L}_{query-group}(x_i, \mathcal{G}; \boldsymbol{\theta}) + \beta \mathcal{L}_{projection}(\mathcal{M}, \mathcal{V}, \mathcal{C}), \tag{8}$$

where α and β is hyper-parameters to scale the three losses and stabilize the training. We set $\alpha = 1.2$ and $\beta = 0.0001$

In inference, we use mDeberta-V3-base (He et al. 2023) to encode the query x_i, select the LLM groups which have the most similarity with the query embedding, and from the selected groups, we can select the optimal LLM which also has the most similarity with the query embedding to answer.

4 EXPERIMENT

4.1 EXPERIMENTAL SETTINGS

For evaluating our routing method, we use 2 benchmark datasets: RouterBench (Hu et al. 2024) and EmbedLLM (Zhuang et al. 2024). These two benchmarks contain a wide range of datasets and the responses from diverse LLMs. We leave two datasets as out-of-distribution(ood) testing, and select some LLMs as our LLM candidates to learn the router and extra several LLMs as new models to simulate the new models' addition procedure as shown in Table 5 and Table 6. The train/validation/test splitting follows the dataset original setting.

Model	Performance↑ non-ood	Performance↑ all	Cost↓ non-ood	Cost↓ all
Ours	**0.69(94.8%)**	**0.67(85.7%)**	**0.58(5.6%)**	**0.80(6.3%)**
Knn	0.50(68.0%)	0.43(54.8%)	0.22(2.2%)	0.31(2.4%)
Bert	0.51(69.4%)	0.43(54.8%)	0.22(2.1%)	0.31(2.4%)
Qwen	0.56(77.1%)	0.56(71.2%)	0.36(3.6%)	0.46(3.6%)
Random	0.56(76.3%)	0.60(76.1%)	3.21(31.4%)	4.11(32.2%)
Oracle	0.85(116.7%)	0.90(113.8%)	1.23(12.1%)	1.48(11.6%)
claude-v2	0.50(68.6%)	0.52(65.5%)	7.52(73.7%)	9.59(75.0%)
gpt-4-1106-preview	0.73(100%)	0.79(100%)	10.2(100.0%)	12.8(100.0%)
code-llama-instruct-34b	0.46(62.7%)	0.51(64.7%)	0.57(5.6%)	0.77(6.0%)
llama-2-70b-chat	0.51(70.7%)	0.56(70.7%)	0.69(6.7%)	0.92(7.2%)
mixtral-8x7b-chat	0.57(78.3%)	0.62(79.4%)	0.45(4.4%)	0.61(4.7%)
Yi-34B-Chat	0.59(80.4%)	0.67(84.9%)	0.61(5.9%)	0.81(6.4%))

Table 1: Main results for RouterBench. Non-ood denotes the results for validation datasets without ood datasets, and all denotes the results for all datasets including the ood datasets. The data in the parentheses means the proportion of the best model `gpt-4-1106-preview`. We select some individual models as a more rigorous comparison.

4.2 BASELINES

To save time, we only compare our methods with 5 common methods: (1) Knn router (Hu et al. 2024): for each test query, the method searches the k neareat samples in the validation dataset and uses the most common used model in these samples; (2) Bert router (Ong et al. 2024): as a classifier problem, we train a Bert model to predict the optimal LLM for each query; (3) Causal LLM router (Ong et al. 2024): instead of using Bert architecture, we use an auto-regressive model Qwen2.5-0.5B (Team 2024) to predict the selected model using the last token's hidden state; (4) Random router: this method randomly sample one LLM to generate the output for each query; (5): Oracle router: oracle means best, which means this router find the optimal LLM for each query, so this method can achieve the highest performance of all routing methods.

4.3 MAIN RESULTS

Main Results. As shown in Table 1, in RouterBench, our method achieves the 94.8% performance of the best single model `gpt-4-1106-preview`, with only 5.6% cost in in-distribution datasets, which is much higher than other routing methods and single models. As for the out-of-distribution(ood) datasets, our methods still 85/7% performance with only 6.3% cost. In Table 2, our method obtains 113.4% accuracy of best model `Llama-3-70B-Instruct` with only 11.1% cost in in-distribution datasets. This digit falls a lot in ood dataset, but it is still higher than other routing methods and single models. For a vivid comparison, you can refer to Figure 3, in which we plot all single models and routing methods. For single dataset results, you can refer to Figure [5, 6, 7, 8].

New Models Previous routing methods usually need retraining the whole system when new model is available, resulting much computational resource waste. Our method can tackle this problem, and the results can be found in Table 3. It is obvious that our method still keeps a high performance than other methods, which demonstrates our method's efficiency and convenience.

Cluster Numbers Actually, the performance depends on the number of LLM groups. In Table 4, we list the performance and cost on different number of LLM groups. For RouterBench, we find that 3 groups can achieve the best performance, and for EmbedLLM, the optimal number of groups is 2. In essence, the more LLMs one each group has, the options for models can be diverse, and extend the search space, which is beneficial to the final result.

Model	Performance↑		Cost↓	
	non-ood	all	non-ood	all
Ours	**0.54(113.4%)**	**0.51(89.8%)**	**247.7(11.1%)**	**295.5(10.8%)**
Knn	0.29(60.1%)	0.27(46.8%)	32.1(1.4%)	45.8(1.7%)
Bert	0.41(86.2%)	0.38(66.7%)	148.6(6.7%)	198.5(7.3%)
Qwen	0.41(86.2%)	0.38(66.7%)	148.9(6.7%)	203.4(7.5%)
Random	0.44(93.1%)	0.42(72.7%)	793.1(35.6%)	997.8(36.6%)
Oracle	0.68(141.9%)	0.82(142.9%)	169.0(7.6%)	225.8(8.3%)
Llama-3-70B-Instruct	0.48(100.0%)	0.57(100.0%)	2229.3(100.0%)	2726.1(100.0%)
Deepseek-67B-chat	0.43(88.5%)	0.49(85.8%)	1859.8(83.3%)	2271.8(83.3%)
Qwen1.5-32B-chat	0.42(88.5%)	0.49(85.8%)	990.8(44.4%)	1211.6(44.4%)
Vicuna-13B-v1.5	0.34(71.2%)	0.39(67.3%)	544.9(24.4%)	666.4(24.4%)
Medicine-LLM	0.28(58.4%)	0.36(62.4%)	297.24(13.3%)	363.48(13.3%)
Gemma-2B-it	0.19(40.4%)	0.23(39.9%)	123.9(5.6%)	151.5(5.6%))

Table 2: Main results for EmbedLLM. The data in the parentheses means the proportion of the best model `Llama-3-70B-Instruct`.

Dataset	Model	Performance↑		Cost↓	
		non-ood	all	non-ood	all
RouterBench	Ours	**0.72(98.9%)**	**0.7(89.0%)**	**0.58(5.7%)**	**0.84(6.6%)**
	Knn	0.48(65.9%)	0.44(55.9%)	0.26(2.5%)	0.35(2.7%)
	Random	0.58(79.6%)	0.62(78.8%)	3.54(34.7%)	4.42(34.6%)
	Oracle	0.89(122.2%)	0.90(114.4%)	1.32(12.9%)	1.58(12.4%)
EmbedLLM	Ours	**0.58(121.3%)**	**0.54(94.3%)**	**298.7(13.4%)**	**395.3(14.5%)**
	Knn	0.3(62.8%)	0.3(52.4%)	78.0(3.5%)	125.4(4.6%)
	Random	0.49(102.5%)	0.45(78.6%)	873.9(39.2%)	1234.9(45.3%)
	Oracle	0.72(150.6%)	0.85(148.4%)	182.8(8.2%)	447.1(16.4%)

Table 3: Main results for new model situation. Because Bert routing and Qwen routing need retraining the system, they can not adapt to the new model's situation.

		2	3	4	5	6	7
EmbedLLM	Perf.↑	0.46	**0.51**	0.46	0.49	0.37	0.41
	Cost↓	454.3	**295.5**	302.9	1211.6	181.7	302.9
RouterBench	Perf.↑	**0.67**	0.65	0.60	0.64	-	-
	Cost↓	**0.80**	0.85	0.73	0.75	-	-

Table 4: Results for different number of LLM groups.

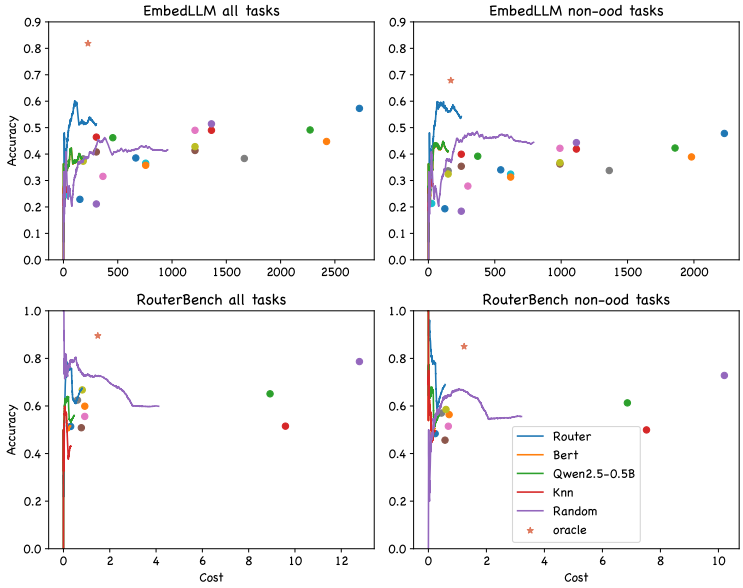

Figure 3: The overall results. The blue line is our method's performance, and the dots in the figures is all single models. The upper two figures are the EmbedLLM results and the bottom two are for RouterBench. The left two figures are for all tasks (including ood tasks), and the right two are for our training tasks.

5 Conclusion

In this work, we explore a two-stage routing approach with embedding representation. We leverage the two contrastive losses to learn the embedding of LLM groups and LLMs, and MSE loss to optimize the projection matrix. Our method achieves competitive performance on in-distribution and out-of-distribution datasets with much lower cost than the best single model and other baselines. In addition, it is convenient to merge new models into out routing system without troublesome retraining.

DIVISION OF TASKS

All teammates contribute to the project equally.

ACKNOWLEDGMENT

See details in https://nlp-course-cuhksz.github.io/.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*, 2024.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

PS Aishwarya, Pranav Ajit Nair, Yashas Samaga BL, Toby James Boyd, Sanjiv Kumar, Prateek Jain, and Praneeth Netrapalli. Tandem transformers for inference efficient llms. In *Forty-first International Conference on Machine Learning*, 2024.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

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

Shuhao Chen, Weisen Jiang, Baijiong Lin, James Kwok, and Yu Zhang. Routerdc: Query-based router by dual contrastive learning for assembling large language models. *Advances in Neural Information Processing Systems*, 37:66305–66328, 2024.

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. *arXiv preprint arXiv:1803.05457*, 2018.

Claude. Claude 3 haiku: our fastest model yet. 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

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

DeepSeek-AI. Deepseek-v3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2023. URL https://arxiv.org/abs/2111.09543.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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. *arXiv preprint arXiv:2403.12031*, 2024.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*, 2021.

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 model routing for efficient llm inference. *arXiv preprint arXiv:2502.08773*, 2025.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *arXiv preprint arXiv:2402.05120*, 2024.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. *arXiv preprint arXiv:2106.15772*, 2021.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

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. *URL https://arxiv. org/abs/2406.18665*, 2024.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pp. 248–260. PMLR, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Marija Šakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 606–615, 2024.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. Mixllm: Dynamic routing in mixed large language models. *arXiv preprint arXiv:2502.18482*, 2025.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning. *arXiv preprint arXiv:2310.03094*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. Capability instruction tuning: A new paradigm for dynamic llm routing, 2025. URL `https://arxiv.org/abs/2502.17282`.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. Embedllm: Learning compact representations of large language models. *arXiv preprint arXiv:2410.02223*, 2024.

A Appendix

A.1 Answer Correctness of EmbedLLM

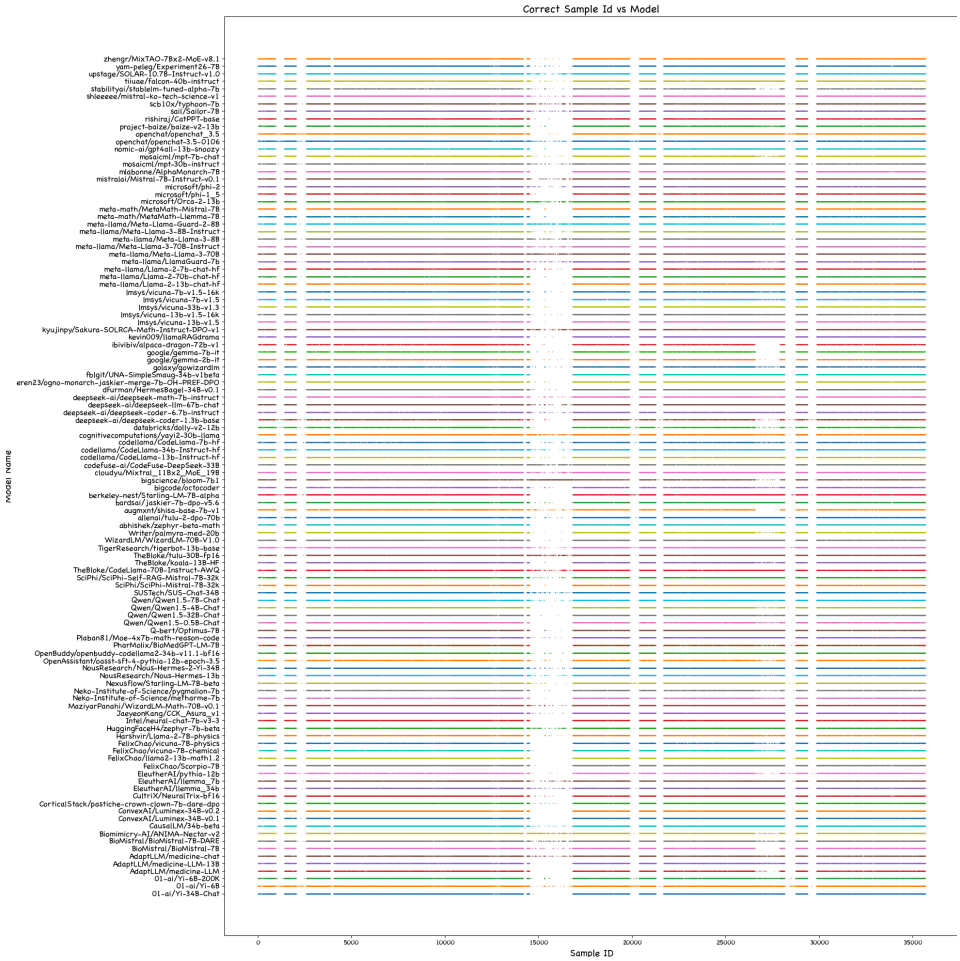

Figure 4: Answer correctness of diverse LLMs on EmbedLLM. The dots in the figure means that the model can solve corresponding data sample. The digits in the left are the accuracies for LLMs.

A.2 Datasets and LLMs

benchmarks	type	datasets
RouterBench	training	Hellaswag (Zellers et al. 2019), MBPP (Austin et al. 2021) MT-Bench (Zheng et al. 2023), MMLU (Hendrycks et al. 2020) GSM8K (Cobbe et al. 2021), Rag
	ood	ARC-C (Clark et al. 2018), Winogrande (Sakaguchi et al. 2021)
EmbedLLM	training	MMLU (Hendrycks et al. 2020), TruthfulQA (Lin et al. 2021) SocialQA (Sap et al. 2019), PIQA (Bisk et al. 2020) GPQA (Rein et al. 2024), MathQA (Amini et al. 2019) LogiQA (Liu et al. 2020), GSM8K (Cobbe et al. 2021)
	ood	ASDiv (Miao et al. 2021) MedMCQA (Pal et al. 2022)

Table 5: Datasets from the two benchmarks

benchmarks	type	models
RouterBench	training	WizardLM/WizardLM-13B-V1.2, claude-instant-v1 claude-v1, claude-v2, gpt-4-1106-preview meta/code-llama-instruct-34b-chat, meta/llama-2-70b-chat mistralai/mixtral-8x7b-chat, zero-one-ai/Yi-34B-Chat
	new	gpt-3.5-turbo-1106, mistralai/mistral-7b-chat
EmbedLLM	training	meta-llama/Meta-Llama-3-70B-Instruct, WizardLM/WizardLM-70B-V1.0 deepseek-ai/deepseek-llm-67b-chat, 01-ai/Yi-34B-Chat fblgit/UNA-SimpleSmaug-34b-v1beta, cognitivecomputations/yayi2-30b-llama Qwen/Qwen1.5-32B-Chat, tiiuae/falcon-40b-instruct Plaban81/Moe-4x7b-math-reason-code, FelixChao/llama2-13b-math1.2 lmsys/vicuna-13b-v1.5-16k, AdaptLLM/medicine-LLM-13B upstage/SOLAR-10.7B-Instruct-v1.0, Nexusflow/Starling-LM-7B-beta google/gemma-7b-it, Qwen/Qwen1.5-7B-Chat AdaptLLM/medicine-LLM, microsoft/phi-2 Qwen/Qwen1.5-4B-Chat, Qwen/Qwen1.5-0.5B-Chat google/gemma-2b-it
	new	EleutherAI/llemma-7b, deepseek-ai/deepseek-math-7b-instruct mosaicml/mpt-30b-instruct, allenai/tulu-2-dpo-70b

Table 6: Models from the two benchmarks

A.3 Single Dataset Results

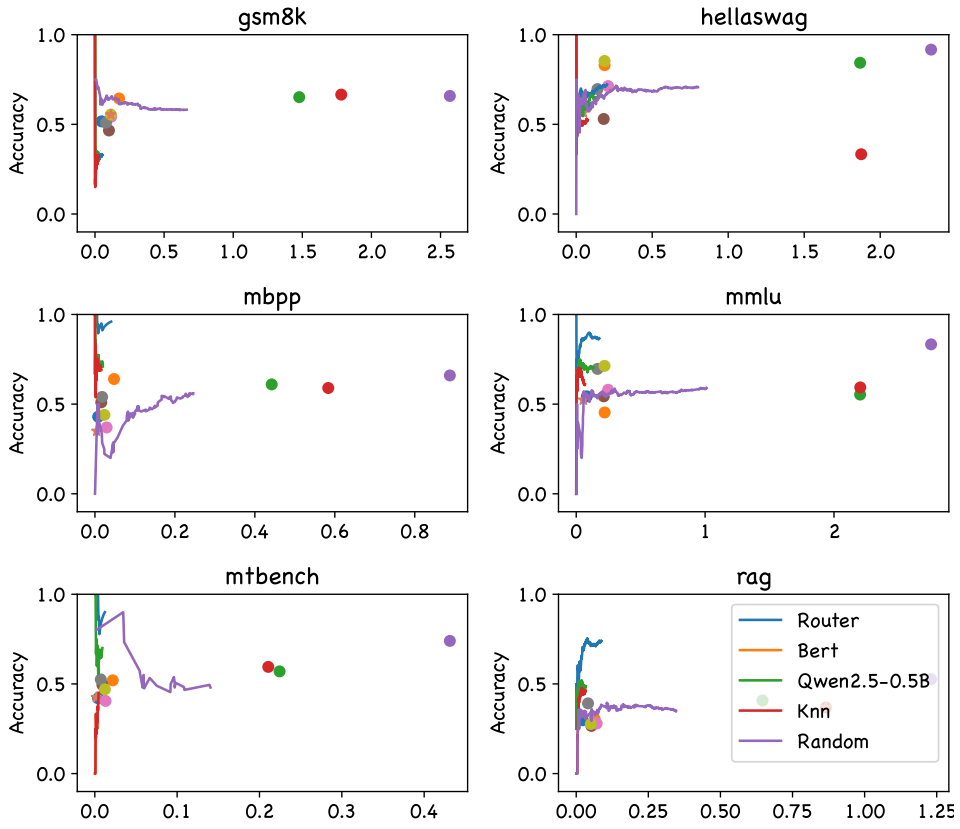

Figure 5: The in-distribution single dataset result for RouterBench.

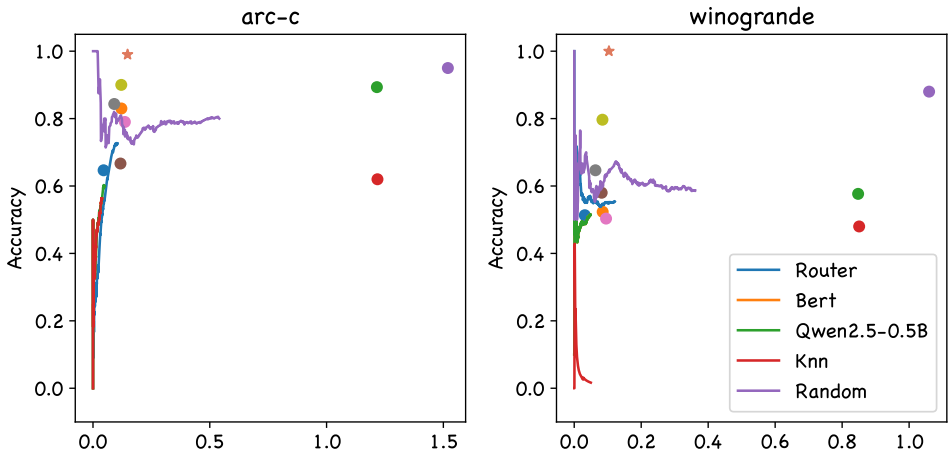

Figure 6: The ood single dataset result for RouterBench.

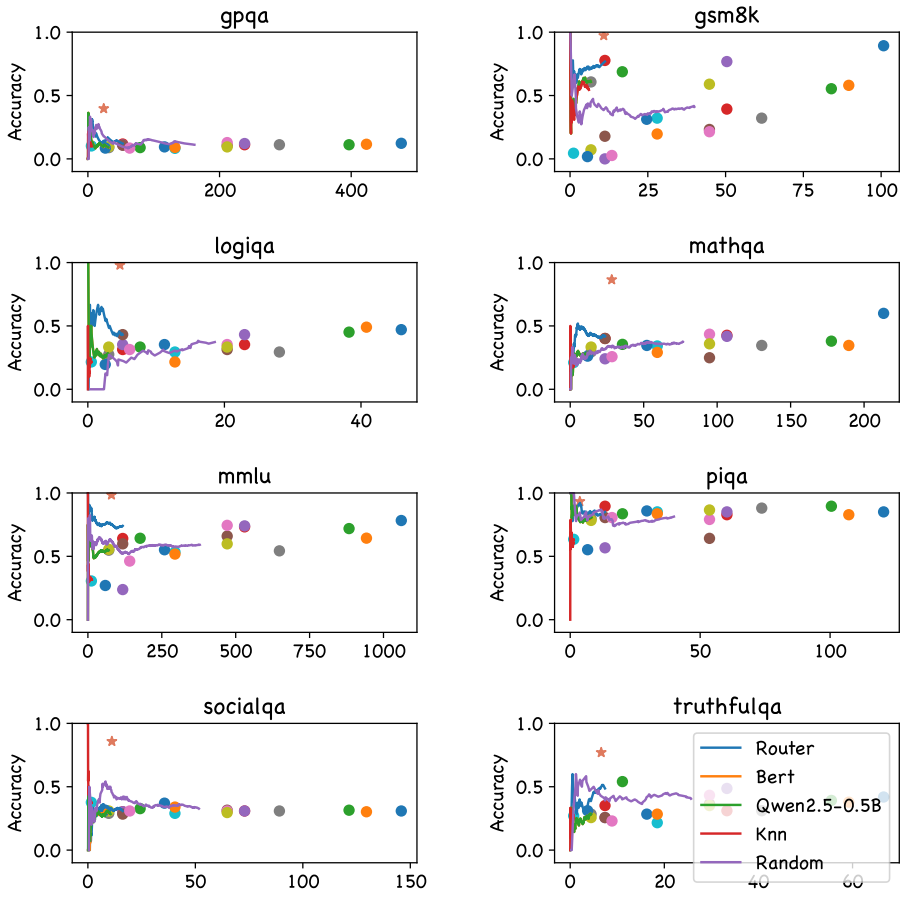

Figure 7: The in-distribution single dataset result for EmbedLLM.

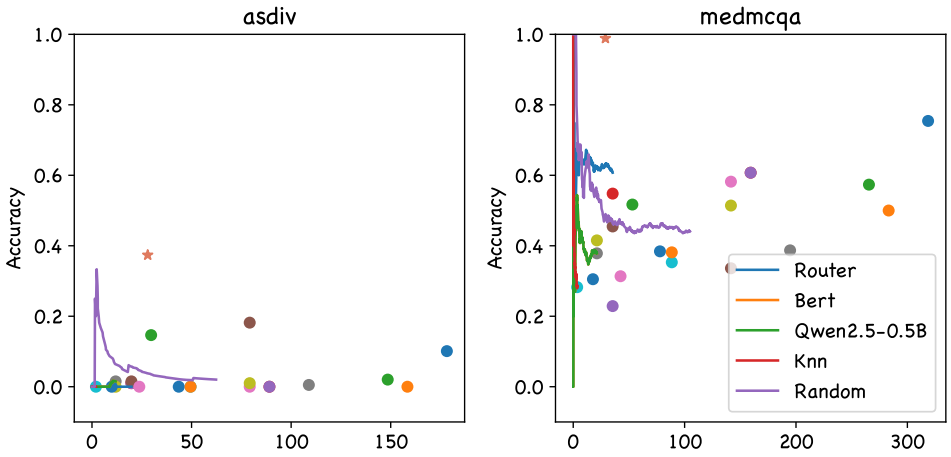

Figure 8: The ood single dataset result for EmbedLLM.