Two-Stage LLM Routing with Embedding Representation

Authors: Zeng Weixuan

School of Data Science, The Chinese University of Hong Kong, Shenzhen, China

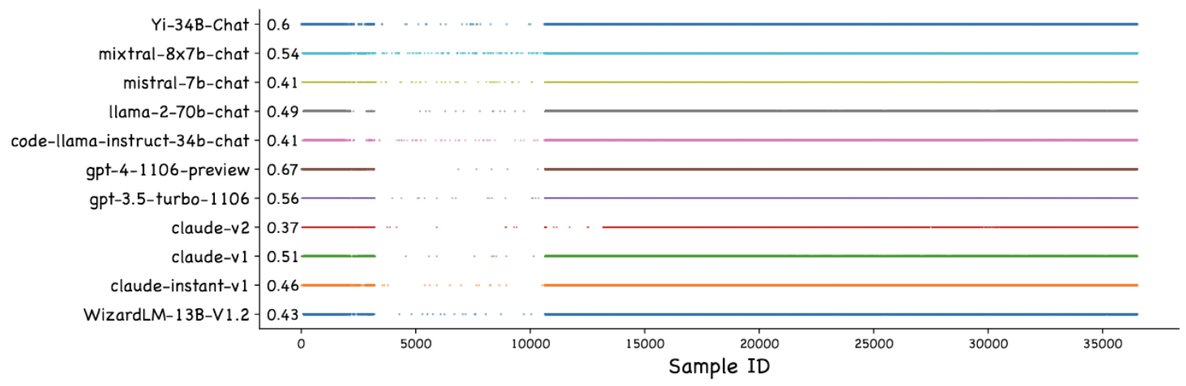
Email: weixuanzeng@link.cuhk.edu.cn

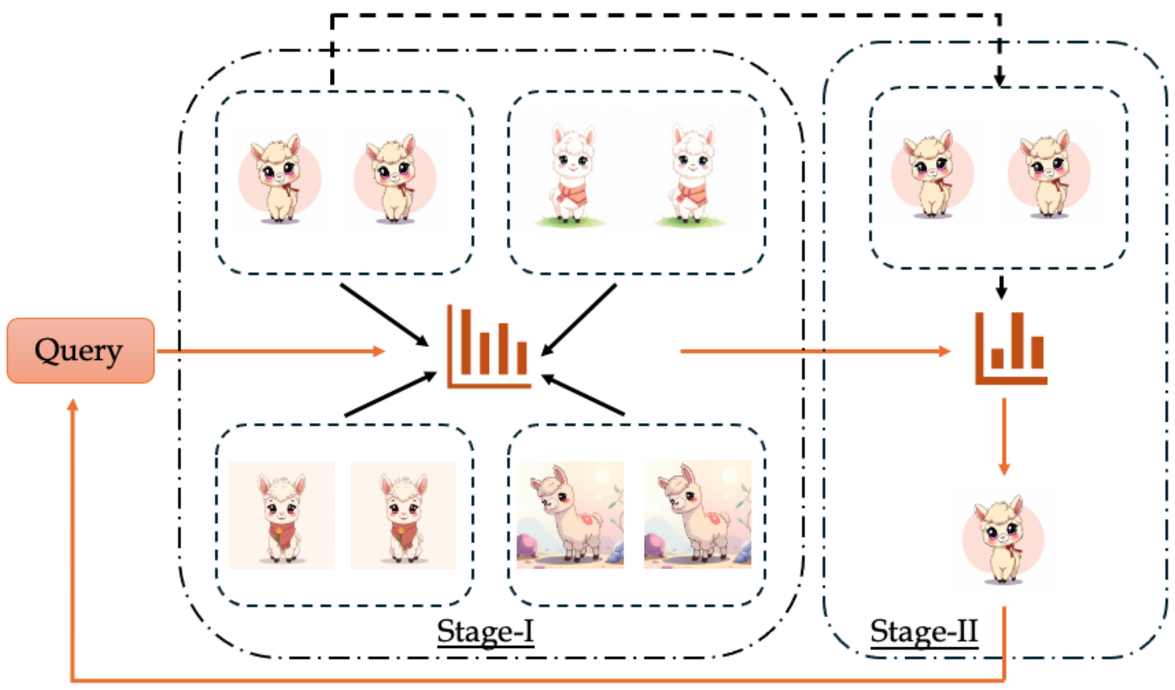
Introduction

In this work, we propose a Two-Stage LLM Routing method via the Embedding Representation.

- LLM routing mainly focuses on how to find an optimal LLM for each query to balance the performance and cost.
- Although larger LLMs generally offer better performance, there is still some problems they can not solve or just answer incorrectly. It is natural to think how about ensembling multiple language models to boost the overall performance. Previous strong baselines can not generalize to new models, when new models become candidates, they need to retrain their models.
- Our method clusters all LLM candidates into several groups based on each LLM's performance and cost. Use two contrastive losses to train the embedding representation of groups and LLMs. To adapt to new models, we also add a MSE loss to link the LLMs' embedding with their performance vectors via a projection matrix.
- In inference, our method first selects the most suitable groups and then select the optimal LLM from the selected groups to compress costs, via the embedding's similarity. When new models come in, we test them in a validation dataset to obtain their performance, then multiply the learned projection matrix to get their embedding representation. Then it is convenient to add new models into our routing system via the embeddings.
- Results show that our 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.

Methodology





Training:
I): a contrastive loss for learning the embedding of each LLM group;
II): a contrastive loss for learning the embedding of each LLM;
III): a MSE loss for optimizing the projection matrix;

Methodology

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

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

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

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

Inference:
After obtaining the embeddings of groups and LLMs via training:
I): compute the similarity of query and the groups to select the groups;
II): compute the similarity of query and the LLMs from the selected groups;
III): leverage the most suitable model from II to generate the response;
Multiply the projection matrix with the performance vectors of new models to obtain their embeddings, thus merging new models.

Results

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%))

Model	Performance↑ non-ood	Performance↑ all	Cost↓ non-ood	Cost↓ 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%))