PROCEED: Performance Routing Optimization for Cost-Efficient and Effective Deployment

Stanford CS224N Custom Project

Lichu Acuña
Department of Mathematics
Department of Computer Science
Stanford University
acuna@stanford.edu

Odin Farkas
Department of Computer Science
Stanford University
omfarkas@stanford.edu

Abstract

The rapid proliferation of Large Language Models (LLMs) necessitates the development of efficient routing systems to optimize model selection while balancing performance and computational cost. Current proprietary LLM routers restrict access to efficient routing technologies, particularly disadvantaging sectors lacking substantial computational resources. To address this, we present an open-source routing framework that predicts LLM performance based on query inputs to determine the most suitable model, thereby minimizing computational overhead while preserving output quality. Utilizing the RouterBench dataset, we leveraged lightweight machine learning models—fastText and DistilBERT—and reformulated the routing problem from classification to regression. These models were chosen for their computational efficiency, a necessary prerequisite for a router to be practical. Our experimental results indicate that both models can distinguish between easy and hard prompts, demonstrating a basic understanding of prompt complexity. Despite challenges in predicting more intricate differences in prompt difficulty, the approach demonstrates great promise for the creation of a predictive solution for LLM routing, with potential for improved accuracy using larger datasets in the future.

1 Key Information

- TA mentor: Yann Dubois
- External collaborators: No
- Sharing project: No
- Team Contributions: Both team members attended weekly meetings with Yann and frequently discussed project direction. Odin led background research and creation of implementation strategies. Lichu led implementation and experiment execution.

2 Introduction

The escalating adoption of Large Language Models (LLMs) across diverse sectors presents a compelling challenge: optimizing the selection of these models in a landscape burgeoning with options. While the allure of deploying the latest, most robust models persists, practical applications often reveal that smaller, less resource-intensive LLMs can suffice, bringing forth significant cost efficiencies. This realization underscores the necessity for intelligent routing systems that not only match the query with the most efficient model but also mitigate the substantial financial impacts associated with high-end models. Recent routing methods, such as LLM cascading, while resourceful, sequentially activate an array of models from the least to the most expensive until they meet a predetermined

output quality. This non-predictive approach, as detailed in works like FrugalGPT (Chen et al., 2023) [1], can lead to excessive computational resource usage without guaranteeing optimal results.

Innovatively, our project pivots from traditional methods by introducing a routing framework that leverages historical data to predict the quality of each LLM output without their actual deployment. Utilizing the RouterBench dataset [2], our approach involves fine-tuning lightweight models to predict the performance levels of LLMs based solely on incoming queries. This strategic shift enhances model selection precision and significantly reduces operational overhead by eliminating the need to test multiple models. Developing a predictive model that accurately forecasts LLM performance presents significant challenges due to the complexity of modeling the nuanced capabilities of diverse LLMs. However, the practical importance of such a model cannot be overstated, as it promises to significantly enhance efficiency in the deployment of LLMs, particularly in large-scale environments.

3 Related Work

Routing In the realm of LLM routing, strategies diverge between enhancing single models and selecting among multiple models without executing each. Techniques like Mixture-of-Experts (MoE) optimize a single model's performance through internal routing to specialized 'experts,' but these are often limited to specific models. Additionally, strategies such as LLM synthesis and non-predictive routing utilize ensemble techniques or direct selection from multiple models' outputs to enhance overall performance without initial predictive assessment. In contrast, predictive routing strategies, exemplified by "Large Language Model Routing with Benchmark Datasets" (Shnitzer et al, 2023) [3], represent a significant advancement. By leveraging historical performance data to predict the optimal model for a given query, these strategies significantly reduce computational overhead and enhance efficiency by obviating the need to generate responses from all models, making them an optimal choice for scalable and effective LLM deployment.

Router Standardization The "ROUTERBENCH: A Benchmark for Multi-LLM Routing System" paper (Hu, Bieker, et al., 2024) [4], addresses the challenge of selecting the most cost-effective large language model (LLM) for specific queries amidst the burgeoning array of LLMs. It introduces the RouterBench dataset, a vast compilation of 405,000 inference outcomes from 11 models across varied tasks, providing a robust platform for both training and benchmarking routing algorithms. This dataset is noted for its diversity, encompassing tasks ranging from commonsense reasoning to mathematical problem-solving, hoping to provide a holistic standard for routing akin to ImageNet's impact on computer vision. Complementing the dataset, the paper presents a novel mathematical framework to evaluate routers by mapping their cost-effectiveness and performance on a c-q plane. This pioneering work establishes a standardized benchmark for router assessment, encouraging advancements in the development of LLM routing algorithms.

Alternative Router Implementations A recent approach to a the predictive routing challenge is explored in the "Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing" paper (Ding et al., 2024) [5].Contrasting with the predictive model proposed in the RouterBench paper, which anticipates the performance of each potential LLM to route queries accordingly, the Hybrid LLM paper advocates a more streamlined strategy. This approach simplifies the prediction to determining the necessity of using a small versus a large model for each task, thus reducing the decision-making process to a choice between two model categories. The authors report that this methodology achieves a 40% reduction in costs without compromising the quality of outcomes, providing a significant enhancement in routing efficiency.

Open Source LLM routing is an emerging field, and many of the solutions, such as those developed by industry leader Martian [6], are proprietary. This situation underscores a significant gap in the development of open-source LLM routing tools.

4 Approach

We approach the routing problem differently from most previous efforts. Instead of developing a single small model that takes a prompt as input and outputs the best LLM to query to optimize performance and cost for that prompt, we build a small model for each LLM. Each of these models takes the prompt as input and outputs the predicted quality score for the response that would be generated by that specific LLM.

In a production environment, the app using our router would query the model for each of the available LLMs to get predicted quality scores $q_1, \cdots, q_n$. A simple deterministic formula would then take these predicted scores into account, along with some preference parameters like the cost/quality trade-off, to choose which LLM to query. For instance, such a formula could have preference parameters $w_{\text{performance}}$ and w_{cost} and look like $w_{\text{performance}} \cdot q_i - w_{\text{cost}} \cdot c_i$. This calculation would be performed for each LLM, and the one with the highest score would be used.

In essence, we do not build a *routing* model but rather multiple *quality prediction* models and then use their outputs to construct a routing algorithm. We choose this approach because:

1. **Ease of Integration and Removal of LLMs**: It simplifies the process of integrating new LLMs into or removing existing LLMs from our routing algorithm. In the single-model approach, the entire model would need to be retrained each time a new LLM is added or removed from the possible options. With our approach, we only need to train a new quality prediction model for any new LLMs added.
2. **Flexibility in Preference Parameters**: It allows for easier modification of the preference parameters for each program using our routing algorithm. In the single-model approach, the entire model would have to be retrained for each different set of preference, such as the cost/quality trade-off parameters shown above. In our approach, these preference parameters are only used in the deterministic formula applied *after* running the quality prediction models. Thus, a single instance of these models can serve all possible set of preferences parameters.

Models We built two different architectures for our quality prediction models.

On the one hand, we use fastText[7] embeddings for our baseline model. FastText embeddings utilize subword information to construct lightweight word embeddings and have also demonstrated good results in embedding entire sentences by averaging the embeddings of each word within them. Our baseline quality prediction model computes the embedding of the prompt and run it through a small neural network with a sigmoid activation at the end, outputting a value between 0 and 1 to predict the quality score of the language model for the given prompt.

Our second model is a fine-tuned extension of DistilBERT[8]. DistilBERT is a lighter version of BERT, with 40% fewer parameters, running 60% faster while preserving 95% of BERT's performance as measured on the GLUE language understanding benchmark. Our model uses DistilBERT up to its last layer, also followed by a small neural network with a sigmoid activation at the end, which outputs a value between 0 and 1. We experimented with training the model using both frozen and unfrozen BERT parameters.

5 Experiments

5.1 Data

The RouterBench dataset is a comprehensive collection designed to support the development and evaluation of routing algorithms for Large Language Models (LLMs). It consists of 36,498 English prompts that cover a wide range of AI-relevant tasks and domains, each associated with responses from various supported LLMs, along with a discrete score for each response, quantifying its quality on a predetermined scale. Notable benchmarks within this dataset include Hellaswag (Zellers et al., 2019), which challenges models to complete realistic, commonsense scenarios requiring an understanding of everyday activities, and MBPP (Austin et al., 2021), which evaluates code synthesis capabilities through Python programming problems. Additionally, the dataset includes a variety of other tasks ranging from conversational benchmarks to complex mathematical problem-solving, further illustrating its capability to provide diverse and rigorous tests for routing algorithms, spanning multiple cognitive and linguistic challenges.

The scoring system in the RouterBench dataset uses discrete values (0, 0.25, 0.5, 0.75, 1), providing a clear, quantitative framework for evaluating the outputs of LLMs. While this quantitative approach is beneficial for straightforward comparisons and systematic analysis, it is important to recognize that the discrete nature of the scoring may not capture the full nuance of model outputs, potentially oversimplifying the complexities of LLM performance.

5.1.1 Removal of Multiple-Choice Prompts

From the 36,498 English prompts in the RouterBench dataset, 27,684 are multiple-choice prompts where the target answer is a single character (usually "A", "B", "C", or "D"). Predicting the quality of a LLM's response to such queries is extremely challenging, as it requires the lightweight model to effectively possess the same knowledge base as the LLM. During initial experimentation, we observed that this negatively impacted the learning of our quality prediction models. For instance, for the GPT-3.5 LLM, the average predicted score by the fine-tuned DistilBERT model for prompts whose responses were scored 0 in RouterBench was 0.5327, while for those scored 1, it was 0.5944. This indicates that the model could barely differentiate between them. Taking this into account, and considering that multiple-choice questions are not representative of the typical prompts that LLMs usually receive, we decided to remove these queries from the dataset. This leaves us with 8,814 prompts in our dataset, which is a relatively small number but sufficient to train a model. Given the dataset's size, we decided to split it into a training set comprising 90% of the dataset and a test set comprising the remaining 10%.

5.1.2 Continuous metric scores

Given the discrete nature of the scores provided by RouterBench, we explored the possibility of integrating a continuous metric to evaluate the quality of the language model's responses. This aimed to differentiate between correct answers by assessing their quality more precisely. While RouterBench assigns a score of 1.0 to any correct response regardless of its quality, a continuous metric could provide a nuanced evaluation, assigning higher scores to higher quality responses.

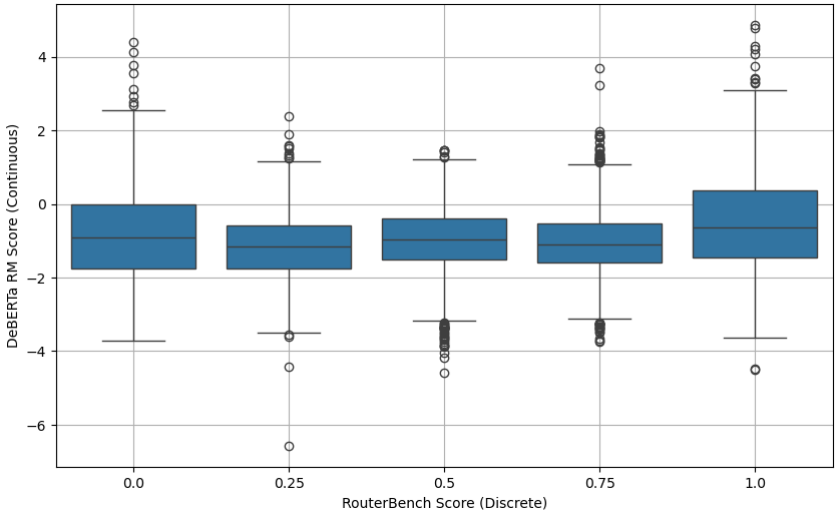

Figure 1: RouterBench Score vs DeBERTa RM Score

To address this challenge, we utilized Microsoft's DeBERTaV3's reward model [9], which was trained during post-training using their Direct Preference Optimization (DPO) strategy [10]. This model is available on HuggingFace through OpenAssistant [11], and to assess its quality we applied it to all responses from GPT-3.5 within the RouterBench dataset.

The scores from DeBERTaV3's reward model, however, showed low correlation with RouterBench scores, with a correlation coefficient of only $\rho = 0.0542$. The following plot illustrates the comparison between the two sets of scores (note that DeBERTaV3's reward model scores are not confined to the $[0, 1]$ interval), showcasing their discrepancies. It can be seen in the plot that DeBERTaV3's reward model assigns only slightly higher scores to responses rated 1 by RouterBench compared to those rated 0, which is undesirable.

Considering that RouterBench scores were assigned in a more controlled and deterministic manner, we regard them as more significant than those assigned by the reward model. Consequently, we decided to use the discrete RouterBench scores for our quality prediction models, instead of the scores generated by DeBERTaV3 reward model.

5.1.3 Dataset Balancing

With these changes, our dataset is composed of:

- 408 prompts with a score of 0.
- 1322 prompts with a score of 0.25.
- 1737 prompts with a score of 0.5.
- 4619 prompts with a score of 0.75.
- 726 prompts with a score of 1.

As we can see, the dataset is heavily imbalanced toward the higher scores. To address this, we perform oversampling in the training set. This involves duplicating rows so that there is an equal number of rows for each score. As a result, the training set consists of 3,735 rows for each of the five score classes.

5.2 Evaluation method

The changes introduced to our dataset and model, specifically the shift from a single-model paradigm to our one-model-per-LLM paradigm, make it challenging to compare our results with those from RouterBench, the only existing benchmark for routing algorithms in the literature. RouterBench introduces and uses a metric called Average Improvement in Quality (AIQ) to assess router performance. However, due to the aforementioned changes in our architecture, it is impossible to calculate this metric in our models and so directly compare our router with theirs. Given the nascent stage of this research area, we are compelled to develop our own evaluation method for our routing algorithm.

First, we examine the loss of our quality prediction model, which uses the Mean Squared Error (MSE) loss. However, to gain a better and more intuitive understanding of what our models are predicting, and leveraging the discrete nature of our dataset, we look at the average predicted score by our model for prompts whose responses received a specific score in the RouterBench dataset. This analysis provides us with five numbers: the average predicted score for the 0 class, 0.25 class, 0.5 class, 0.75 class, and 1 class. We particularly focus on the average predicted quality for the 0 class being significantly different from the average predicted quality for the 1 class, as these two are the most distinct. A considerable distance between these averages indicates that our models are effectively predicting the difficulty of a given prompt. To visualize these results, we plot the comparison between RouterBench and our predicted values in a box plot, similar to what we did in Figure 1.

5.3 Experimental details

Both the fastText-based model and the DistilBERT-based model were trained using the Mean Squared Error (MSE) loss function.

We conducted a grid search to find the optimal architecture for our fastText and DistilBERT models. The hyperparameters we searched over during our grid search included:

1. Number of neurons per layer: 16, 32 or 64, 128, 256.
2. Number of layers: 1, 2, 3, 4 or 5.
3. Learning rate: 0.0001, 0.001, 0.01 or 0.1
4. Number of epochs: 1, 2, 5, 10 or 20.
5. Batch size: 32, 64, 128, 256.

Note that the hyperparameters related to the architecture of the model (1 and 2) affect what runs *after* the fastText embeddings or the DistilBERT model.

5.3.1 fastText-Based Model

While fastText allows for training embeddings on your own data, we decided to use the pre-trained English word vectors [12] since they were trained on a significantly larger dataset and are therefore richer. These embeddings are of dimension 300, but we used fastText's native functionality to reduce them to 50 dimensions to make our model lighter.

The grid search resulted in the following optimal architecture for the fastText-based model:

1. Number of neurons per layer: 64
2. Number of layers: 1
3. Learning rate: 0.001
4. Number of epochs: 10
5. Batch size: 64

As a note, we also experimented with incorporating the length of the prompt as an input to our neural network, in addition to the 50-dimensional sentence embedding. To our surprise, however, this did not provide any improvement in the loss of our model. For any hyperparameter configuration, the results were essentially the same as when the length of the prompt was not included in the input.

5.3.2 DistilBERT-Based Model

We train the DistilBERT-based model with the DistilBERT weights frozen.

The grid search resulted in the following optimal architecture for the DistilBERT-based model:

1. Number of neurons per layer: 16
2. Number of layers: 1
3. Learning rate: 0.01
4. Number of epochs: 1
5. Batch size: 128

Refer to Figure 4 in the Appendix for the loss plot of this model's training.

Training with unfrozen DistilBERT weights was also considered. We experimented with re-training the model with unfrozen DistilBERT weights and a smaller learning rate of 0.005 after finishing the first pass of training with frozen DistilBERT weights. However, this did not lead to any decrease in the loss. We believe this is due to the relatively large size of DistilBERT compared to the small size of our data set. We decided, consequently, to keep the DistilBERT weights frozen, given that they are the result of training over multiple orders of magnitude more data and therefore are rich in knowledge. Refer to Figure 5 in the Appendix to see the evolution of the loss when training with unfrozen weights after training with frozen weights.

5.4 Results

While we train the quality prediction model for all LLMs, the results reported here are specifically for training on the scores using the GPT-3.5 model. These results are generally consistent across the different LLMs.

Using the hyper-parameter configurations described in the previous section, we obtained the following results:

	fastText-Based	**DistilBERT-Based**
Training loss	0.0916	0.1029
Test loss	0.0601	0.0587
Avg. score for 0-rated prompts from test set	0.3834	0.4472
Avg. score for 0.25-rated prompts from test set	0.4945	0.5545
Avg. score for 0.5-rated prompts from test set	0.4952	0.5539
Avg. score for 0.75-rated prompts from test set	0.4927	0.5538
Avg. score for 1-rated prompts from test set	0.6278	0.7195

We observe that the difference between the average scores for 0-rated prompts and 1-rated prompts is 0.24 for the fastText-based model and 0.28 for the DistilBERT-based model. This indicates that both models can effectively distinguish between easy and hard prompts. Additionally, the DistilBERT-based model achieves a better loss than the fastText-based model, although the difference is not substantial. While we expected a more significant performance difference between the fastText-based and DistilBERT-based models, this could be due to the fact that the DistilBERT-based model was trained with its weights frozen, essentially making DistilBERT a powerful sentence encoder. With

more extensive and diverse data, the DistilBERT weights could be unfrozen during training, potentially improving the performance of the DistilBERT-based quality predictor model and increasing the gap between it and the fastText-based model.

The fact that the training loss is lower than the test loss can likely be attributed to the oversampling done to balance the data, which was only performed in the training set. We did not observe this phenomenon before balancing the dataset.

The following box plots show the relationship between scores provided by RouterBench and those predicted by our models.

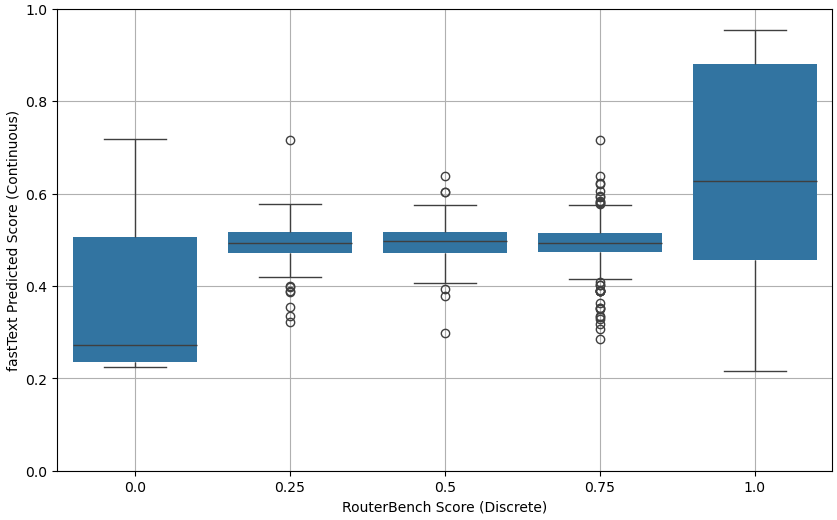

Figure 2: RouterBench Score vs fastText Predicted Score

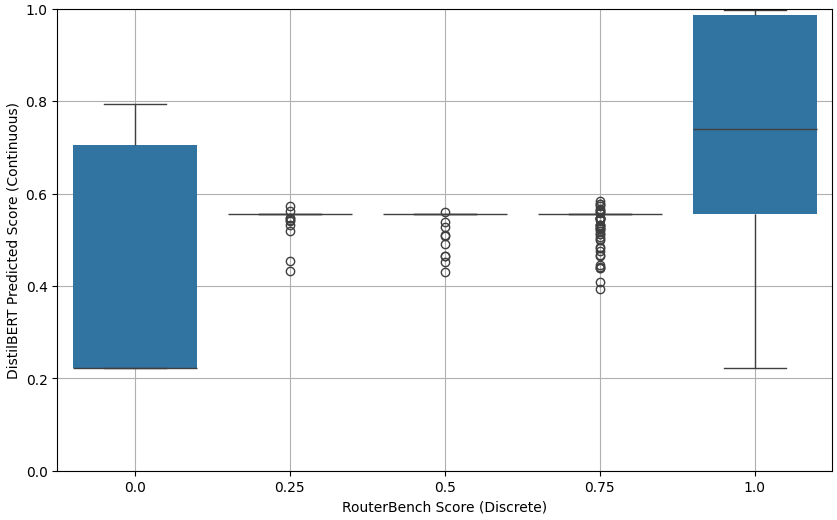

Figure 3: RouterBench Score vs DistilBERT Predicted Score

6 Analysis

Our quality prediction model has shown that it can distinguish between easy and hard prompts, reflecting some degree of understanding of prompt complexity. However, it encounters challenges in discerning nuanced differences in similar prompt difficulties and tends to be overly optimistic,

often predicting higher performance scores than those actually observed, even after oversampling the smaller classes. This optimistic bias indicates that while the model grasps the general quality of responses, it struggles to accurately predict the precise level of performance, particularly in distinguishing subtle differences in quality among high-scoring outputs. The limited training set, comprised of less than 10,000 entries from the RouterBench dataset, likely contributes to this discrepancy, suggesting that a more expansive dataset could potentially enhance prediction accuracy.

We also observe how fastText and DistilBERT perform relatively similarly. Our interpretation for this is that they reach . In future work, a bigger dataset could not only contribute better generalization, but also allow the DistilBERT model to be trained with unfrozen weights, which we were unable to do due to the size of our dataset.

While the model shows promise, there is a clear need for accuracy enhancements to meet the practical demands of LLM routing. Initial thoughts suggested that larger models might improve performance, however empirical evidence from our experiments suggests otherwise. This finding aligns with necessary constraints of our project, as deploying significantly larger models would contravene our goal of maintaining a lightweight, efficient routing system. The need for lightweight models in routing stems from the necessity for quick and efficient performance evaluations, which is critical in operational environments where speed and resource efficiency are paramount. Our findings underscore a fundamental tension in predictive routing: the desire for lightweight, fast models that can quickly provide routing decisions conflicts with the goal of achieving the highest possible prediction accuracy. This tension highlights the intricate balance required in designing routing systems for large language models, emphasizing why deploying larger pretrained models is not an ideal solution.

7 Conclusion

Our project successfully establishes the viability of a predictive routing framework that dynamically selects language models based on performance metrics in real-time. Both the fastText and DistilBERT-based models demonstrated proficiency in differentiating between easy and hard prompts, showcasing a solid foundational understanding of prompt complexity. Although the models exhibited optimistic biases, predicting higher performance scores than observed, these biases underscore areas ripe for further refinement. Particularly, the DistilBERT-based model showed promising results even with frozen weights, indicating potential for enhanced performance with larger datasets and fine-tuning. This framework not only provides an efficient alternative to proprietary LLM routing solutions but also sets the stage for future developments aimed at improving prediction accuracy and computational efficiency. Our approach paves the way for democratizing access to advanced LLM deployment strategies, with future iterations likely to deliver even more robust and precise routing capabilities.

The effectiveness of our routing solution is highly dependent on the diversity and scope of the RouterBench dataset, which, despite its breadth, remains relatively limited in size. This reliance potentially restricts the generalizability of our models across broader real-world applications. Furthermore, our commitment to maintaining a lightweight routing system, essential for rapid and efficient model selection, inherently places constraints on the accuracy and depth of predictions our models can offer. These limitations underline the challenges of balancing performance with practical deployment needs in LLM routing.

While training on a larger dataset is an obvious future direction, another key area of focus is expanding our modular routing solution composed of multiple independent quality prediction models with the integration of new LLMs, such as GPT-4o and beyond. Future enhancements could focus on expanding the range of models supported and refining the integration process to maintain ease of use and adaptability. Additionally, the development of an API for deploying our routing models in production environments represents a crucial step towards practical application, enabling users to leverage our routing system seamlessly in real-time applications. These advancements would further the capability of our routing framework, making advanced LLM routing more accessible and customizable to user needs.

PROCEED.

8 Ethics Statement

Our project faces ethical challenges and societal risks in the areas of bias in model selection and alignment of routing decisions with user expectations concerning data sensitivity. The risk of bias emerges if the routing algorithm fails to properly account for the cultural or demographic specifics embedded within queries, potentially leading queries to models ill-equipped for their nuances, thus perpetuating biases or yielding culturally inappropriate responses. Additionally, if routing decisions neglect the context or sensitivity of queries, especially those containing sensitive information like health data or personal advice, it could lead to breaches of user trust and privacy concerns.

To counteract these risks, we could enhance the RouterBench dataset's utility through a detailed analysis and careful adjustment of our model selection criteria. Recognizing the dataset's limited demographic and cultural representation, we could to develop a context-aware routing protocol capable of intelligently interpreting the subtleties within each query. This could involve integrating external metadata that adds cultural or demographic indicators to each query, aiding in more accurate model selection. Regarding the misalignment with user expectations, we propose establishing strict compliance checks and security protocols to ensure all models within the routing system adhere to robust standards of data confidentiality and user privacy. Complementing this with transparent communication about how user data is handled and which models are used for processing their queries will enhance trust and align our practices with user privacy expectations, thus fostering a responsible deployment of this technology.

References

[1] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. 2023.

[2] Martian AI. RouterBench Dataset. `https://huggingface.co/datasets/withmartian/routerbench`, 2024.

[3] Tal Shnitzer, Anthony Ou, et al. Large language model routing with benchmark datasets. 2023.

[4] Qitian Jason Hu et al. Routerbench: A benchmark for multi-llm routing system. *ICLR AGI Workshop*, 2024.

[5] Dujian Dinga, Ankur Mallick, et al. Hybrid llm: Cost-efficient and quality-aware query routing. 2024.

[6] Martian AI. Martian model router. `https://withmartian.com/products/model-router`.

[7] Facebook AI Research. fasttext: English word vectors. `https://fasttext.cc/docs/en/english-vectors.html/`.

[8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. `https://arxiv.org/pdf/1910.01108`, 2019.

[9] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. 2021.

[10] Rafael Rafailov, Archit Sharma, et al. Direct preference optimization: Your language model is secretly a reward model. 2023.

[11] Hugging Face. Openassistant reward model - deberta v3 large v2. `https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2/tree/main`.

[12] Facebook AI Research. Word vectors for 157 languages. `https://fasttext.cc/docs/en/crawl-vectors.html`, 2021.

A Appendix

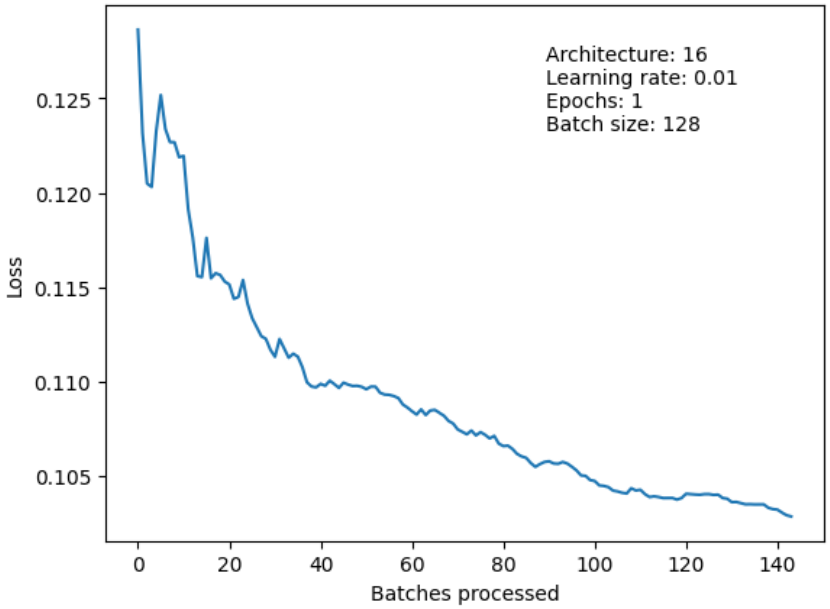

Figure 4: Loss vs Batches Processed for the DistilBERT-based model

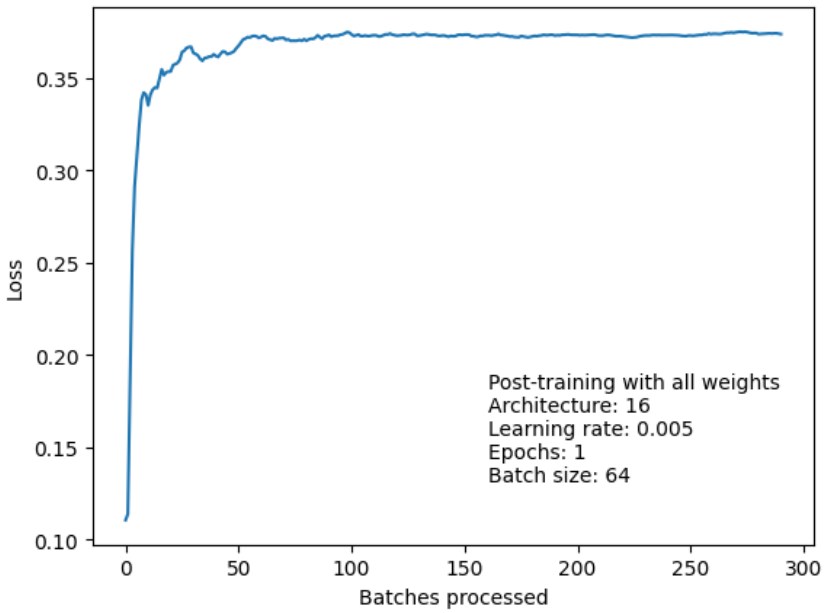

Figure 5: Loss vs Batches Processed for the post-training with unfrozen weights