PickLLM: Context-Aware RL-Assisted Large Language Model Routing

Dimitrios Sikeridis, Dennis Ramdass, Pranay Pareek
{dimitrios.sikeridis,dennis.ramdass,pranay.pareek}@broadcom.com
AI and Advanced Services, VMware Cloud Foundation (VCF) Division, Broadcom
Palo Alto, California, USA

Abstract

Recently, the number of off-the-shelf Large Language Models (LLMs) has exploded with many open-source options. This creates a diverse landscape regarding both serving options (e.g., inference on local hardware vs remote LLM APIs) and model heterogeneous expertise. However, it is hard for the user to efficiently optimize considering operational cost (pricing structures, expensive LLMs-as-a-service for large querying volumes), efficiency, or even per-case specific measures such as response accuracy, bias, or toxicity. Also, existing LLM routing solutions focus mainly on cost reduction, with response accuracy optimizations relying on non-generalizable supervised training, and ensemble approaches necessitating output computation for every considered LLM candidate. In this work, we tackle the challenge of selecting the optimal LLM from a model pool for specific queries with customizable objectives. We propose PickLLM, a lightweight framework that relies on Reinforcement Learning (RL) to route on-the-fly queries to available models. We introduce a weighted reward function that considers per-query cost, inference latency, and model response accuracy by a customizable scoring function. Regarding the learning algorithms, we explore two alternatives: PickLLM router acting as a learning automaton that utilizes gradient ascent to select a specific LLM, or utilizing stateless Q-learning to explore the set of LLMs and perform selection with a ϵ-greedy approach. The algorithm converges to a single LLM for the remaining session queries. To evaluate, we utilize a pool of four LLMs and benchmark prompt-response datasets with different contexts. A separate scoring function is assessing response accuracy during the experiment. We demonstrate the speed of convergence for different learning rates and improvement in hard metrics such as cost per querying session and overall response latency.

1 Introduction

Recently, generative Large Language Models (LLMs) have emerged as a clear trend for solving a diverse set of problems such as assistive dialogue, summarization, text classification and coding aid [2, 24]. In addition, they are becoming more and more capable in terms of understanding complex contexts and providing factual accuracy due to larger training datasets, and an increase in model sizes [27]. Thus, their applicability for problem-solving across diverse applications and industry domains has led to the explosion of their number with new LLMs being released daily by academia and industry alike[1], both in open-source fashion (e.g., Meta's Llama [25]) but also as fully supported LLMs-as-a-service(e.g., OpenAI's GPT models [1]).

While the observed influx of proprietary and open-source LLMs adds diversity and a plethora of options for practitioners and newly flocked users, there are no obvious selection criteria for picking between the different LLM offerings [12]. Indeed, open-source LLMs tend to exhibit diverse weaknesses, strengths, and heterogeneous expertise in various domains mainly due to variations in architecture, training data input, and parameter tuning [2, 34]. Still, interested users are left with a couple of options to adjust LLMs to their use cases including fine-tuning, and even training their LLM from scratch. The latter can be an expensive process in terms of computation and large volume data gathering, and even challenging to even perform due to the scarcity of GPUs in the market. Fine-tuning on the other hand is resource friendlier, but requires a specific level of expertise among engineering teams. In addition, modern LLM offerings often restrict any direct model optimization through weight adjustments, providing just access to black-box APIs.

For all the reasons above, more and more practitioners are now relying on off-the-shelf LLMs for building their applications. However, relevant concerns are surfacing regarding the practical aspects of running them, namely rising costs, availability, inference latency, and average accuracy depending on the use case [3, 16, 32]. Indeed, the latest state-of-the-art models comprise billions of parameters leading to excessive computing power needs and even environmental impact, especially in cases where the application demands high-throughput delivery. A case in point is OpenAI's ChatGPT where the same query (4K-token context) is ≈20 times cheaper when GPT-3.5 is used in comparison to the newer GPT-4 model (8K-token context)[2]. Interestingly, the estimated accumulated cost of running a small business's customer support service in GPT4 can exceed $21K per month [3].

[1] e.g., Hugging Face currently hosts an impressive number of models related to text generation
[2] https://openai.com/pricing

While such cutting-edge models can indeed handle the most demanding text generation tasks, the reality is that many use cases could be handled more than adequately by less-capable and thus less-expensive models. In addition, depending on the use case, the use of local open-source models or even quantized models running on CPUs can eliminate network latency and availability concerns where accuracy degradation can be tolerated. Given the observed variations in heterogeneous cost and quality, the efficient capitalization and real-time evaluation of all the available LLM solutions is quite understudied. There is, thus, the opportunity to identify the optimal LLM for a specific task (i.e., family of queries) and within a desired context (e.g., availability, latency, cost) in a lightweight, real-time manner that can also be adaptable to changes in query themes on-the-fly.

Contributions. In this paper, we tackle the challenge of selecting the optimal LLM from a model pool for specific queries with customizable objectives. Our contributions are summarized as follows:

- We propose PickLLM, a lightweight framework that relies on Reinforcement Learning (RL) to route on-the-fly queries to available models.
- We introduce a weighted reward function that considers per-query cost, inference latency, and model response accuracy by a scoring function.
- We explore two alternatives: PickLLM router acting as a learning automaton that utilizes gradient ascent to select a specific LLM, or PickLLM utilizing stateless Q-learning to explore the set of LLMs and perform selection with a ϵ-greedy approach.
- We demonstrate the speed of convergence for different learning rates, and the effect of variations on the reward weights that result in selecting the optimal LLM in terms of cost, and latency with no accuracy degradation.

Structure. Section 2 reviews related work on LLM selectors and routing. Section 3 analyses the PickLLM framework, and the underlying RL mechanisms. Section 4 describes our experimental setup, and presents our findings. Finally, Section 5 discussed future directions, while Section 6 concludes this paper.

2 Related Work

Existing solutions attempt optimizing LLM selection based on either strict monetary cost, or answer accuracy, rarely for both, and certainly without taking into account other throughput performance metrics. In [3], the authors propose the sequential use of different increasingly expensive LLMs in a cascade manner until the output of a dedicated scoring model (specifically DistilBERT) is acceptable. Similarly, AutoMix [16] utilizes small-size LLMs to produce an approximate correctness metric for outputs before strategically routing the queries further to larger LLMs. LLM cascades are also used in [31], where the authors utilize the "answer consistency" [26] of the weaker LLM as the signal of the query difficulty that will drive the decision to route towards a more capable, and more expensive LLM. The requirement of the above solutions falls on potentially utilizing multiple models per input before yielding the optimal result.

Another family of solutions relies only on computationally heavy pre-training of supervised reward models or uses generic datasets for the supervised training which does not provide generalization for different user cases. For instance, the authors in [21] utilize a regression model that is pre-trained on pairs of language model queries and related output scores. The work in [22] utilizes existing benchmark datasets to train an LLM router model through a collection of binary classification tasks. In [15] the authors deploy reward model ranking on a query-answer training set to obtain an estimation of expertise between open-source LLMs. The normalized rewards from the training phase are subsequently used to train a routing function that will make the final routing decision of a new query in the online phase.

Finally, prior work on LLM selection relies on the selection of the LLM that generates the optimal output for the given input after producing all the possible outcomes. The authors in [14], and [20] propose the training of specific ranking and scoring models after considering outputs from every examined LLM. The work in [10] introduces a pairwise comparison to evaluate differences between candidate LLM outputs from the same query. Following that, a fusing mechanism merges the candidate responses that ranked higher to create the final improved output. However, ranking between answer pairs, although effective, becomes infeasible for real-time applications as the LLM option space expands as each candidate has to be compared with all the rest.

Recently, a promising solution to the LLM routing problem has been proposed in [19]. The framework involves learning to dynamically route between a stronger model and a weaker model, thereby minimizing costs while achieving a specific performance target. The authors utilize human preference data and data augmentation techniques to enhance performance, and demonstrate significant cost savings without compromising response quality. This work highlights the potential of LLM routing to provide a cost-effective yet high-performance solution for deploying LLMs.

3 PickLLM

3.1 Overall Framework

In this section, we describe the basic mechanisms behind PickLLM, which specifically addresses the problem of selecting the optimal LLM for a single use case by taking into account contextual optimization choices such as running cost, or response latency. We consider a pool of $|M|$ available LLMs (set $M = \{1, ..., n, ..., |M|\}$) ready to receive user queries. The LLM set can be diverse, consisting of a mix of

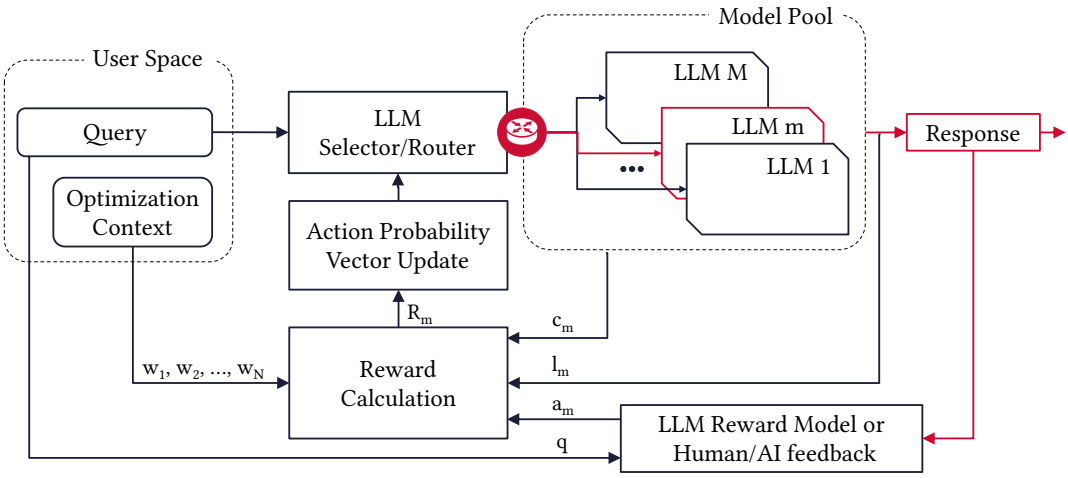

Figure 1. SLA-based PickLLM framework.

local quantized models, local models running on GPUs, or off-the-self LLMs served in cloud platforms. For each round, PickLLM routes the user query q to one of the available models, observes the outcome of each response about multiple metrics, and updates the underlying LLM selection probabilities depending on the exact learning algorithms used. For each user session, i.e., a set of contextually related queries, PickLLM will converge to the most appropriate model given the optimization choices. Fig. 1 presents an overview of the proposed framework.

3.2 Reward Function Formulation

Regarding the reward function, we considered a small set of hard performance-related parameters that are critical for the user's experience and perceived satisfaction. For each model m, we define, (a) the cost per query c_m, associated with the cost of running the model either locally or on the cloud, (b) the inference latency l_m, namely the observed inference runtime (can also account for each model's service options, variations in utilized hardware, software optimizations, and requests spending times in queues), and (c) a response accuracy metric a_m, defined as the average correctness of the LLM's answer based either on human feedback or on a mixture of scoring functions. Given that, he reward function R_m is defined as follows:

$$R_m(a_m, c_m, l_m) = \frac{w_a \cdot a_m - w_c \cdot c_m}{w_l \cdot \frac{\log_{10}(l_m)}{t_{scaling}}} \tag{1}$$

where w_a, w_c, and w_l are the weights assigned to accuracy, cost, and latency, and $t_{scaling}$ is a constant used to scale latency depending on the unit of time used. Evidently, the reward reflects the "competitiveness" of each LLM n as formed by response performance and user preferences. Note, that the proposed reward function is easily extensible to accommodate other optimization targets e.g., bias, toxicity, environmental impact, and misinformation, among others. HELM [13] offers an exhaustive list or possible LLM evaluation metrics.

3.3 Gradient Ascent Learning

First, we examine a gradient ascent learning approach where PickLLM acts as a stochastic learning automaton (SLA) the learns the environment by performing updates of the perceived reward [18]. PickLLM maintains an action probability vector $\mathbf{P}^{[i]} = [P_1^{[i]}, \ldots, P_m^{[i]}, \ldots, P_{|M|}^{[i]}]$, where $P_m^{[i]}$ represents the probability of PickLLM router selecting LLM m for handling the upcoming query at iteration i. The adjustment of action probabilities follows the Linear Reward-Inaction (LRI) algorithm: when the current selection strategy is $\sigma = m$, the probability of continuing using the same LLM m is updated as:

$$P_m^{[i]} = P_m^{[i-1]} + \beta \cdot R_m^{[i-1]} \cdot (1 - P_m^{[i-1]}) \tag{2}$$

while the probabilities of each one of the rest LLMs m' is:

$$P_{m'}^{[i]} = P_{m'}^{[i-1]} - \beta \cdot R_m^{[i-1]} \cdot P_{m'}^{[i-1]}, \forall\, m' \in M, m' \neq m \tag{3}$$

where $\beta \in (0, 1]$ denotes the learning rate, and controls the convergence of the process. PickLLM naturally converges to a single LLM for the session of queries when the action probabilities vector delta is smaller a predefined threshold ϵ: i.e., $\max_\sigma |\Delta P_m^{[i]}| < \epsilon$ for all strategies/choices m, where ϵ is a small positive value indicating the threshold for negligible updates, ensuring that the decision-making process has stabilized. Fig. 1 presents an overview of the proposed framework.

3.4 Stateless ϵ-greedy Q-Learning

In addition, we study stateless ϵ-greedy Q-Learning approach as an alternative mechanism for PickLLM [6, 23, 28]. Under this scenario PickLLM maintains an action-value vector

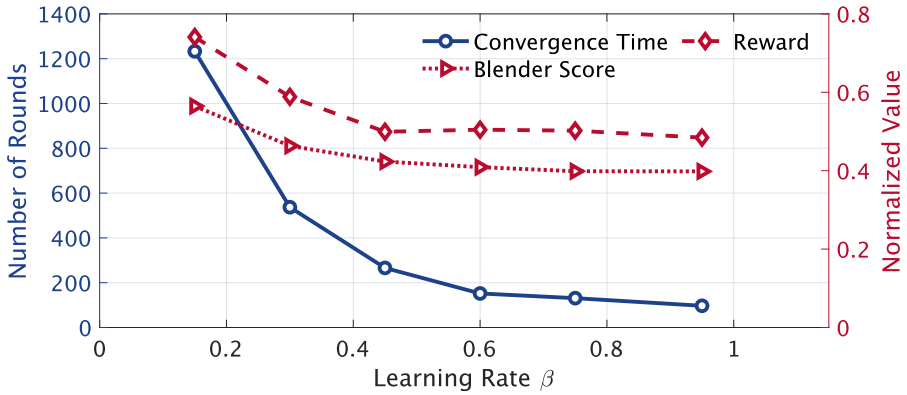

Figure 2. Average of convergence time, reward and LLM-Blender score vs learning rate β

$\mathbf{Q}^{[i]} = [Q_1^{[i]}, \ldots, Q_m^{[i]}, \ldots, Q_{|M|}^{[i]}]$, where $Q_m^{[i]}$ approximates the utility of selecting LLM m up to iteration i, symbolizing the expected reward $R_m^{[i]}$ given the selection of LLM m, namely strategy $\sigma = m$: $Q_m^{[i]} \approx E[R_m^{[i]} | \sigma = m | i]$. The adaptation of action values follows the Q-Learning update rule:

$$Q_m^{[i]} = Q_m^{[i-1]} + \theta \cdot \left(R_m^{[i-1]} - Q_m^{[i-1]} \right) \tag{4}$$

where $\theta \in (0, 1]$ is the learning rate, influencing the magnitude of updates. For selecting LLMs, we employ an ϵ-greedy approach, where PickLLM chooses the strategy σ that maximizes the expected reward as follows:

$$\sigma^{[i]} = \begin{cases} \arg\max_{m \in M} \mathbf{Q}^{[i-1]}, & \text{with probability } 1 - \epsilon \\ \text{random LLM } m \text{ from } M, & \text{with probability } \epsilon \end{cases}$$

allowing PickLLM to explore non-maximal strategies with a probability ϵ, thereby balancing exploration and exploitation.

4 Performance Evaluation

4.1 Experimental Setup

The experimental testbed consisted of a local host and remote execution for the LLMs utilizing an internal in-house LLM API Service. The local client running PickLLM was equipped with an Intel i9 with eight cores at 2.4 GHz each, and 32 GB of RAM.

We used four models to construct our set $|M|$, and specifically Mistral-7B[3] [8], WizardLM-70B[4] [29], and two versions of Llama-2 [25], one with 13 Billion parameters[5], and one with 70 Billion[6]. For our experiments' dataset we utilize the HC3 dataset [5] that consists of a mix of questions of varying topics and their corresponding human answers. HC3 is a collection of different individual datasets, with domains that include open-questions [4, 30], finance [17], and medicine [7].

To simulate cost-related impact, we assume that the use of each model is associated with a normalized cost per query c_m, and define a cost vector $\mathbf{C_{exp}} = [0.4, 0.8, 0.7, 0.3]$ corresponding to Mistral-7B, WizardLM-70B, Llama2-70B, Llama2-13B, chosen to simulate their inference cost and diversify the selection space. As our latency metric l_m we define the duration of an end-to-end communication between a user making a query and getting the response, measured in milliseconds. Finally, regarding response quality/accuracy metric, we utilize two alternatives during our experiments, namely (a) the Open Assistant's [11] deberta-v3-large-v2 reward model [7], and (b) OpenAI's gpt-3.5-turbo[8] acting as a response quality judge [33] see specific system prompt in Appendix A. Both of them are normalized to yield a score $a_m \in (0, 1]$. Since the datasets employed in our experiments include human-generated answers for each query, we also utilize LLM-Blender's [10] pairwise reward model (PairRM)[9] that produces a relative quality score between two response candidates given a specific input query.

4.2 Evaluation and Discussion

First, we consider how PickLLM learning rates impact the action probability convergence speed and quality of learning. Our experiment utilizes the ELI5 [4] portion of the test dataset, fixes the reward weights to $w_a = 0.5$, $w_c = 0.25$, $w_l = 0.25$, and averages 10-runs for different values of the SLA learning rate β. Fig. 2 shows the average convergence time in terms of querying rounds for increasing values of β. For larger values of β the convergence time is lower with a reduction of $\approx$ 82% when moving from 0.3 to 0.9. However, for small β values the system usually concludes with better choices. In this context, Fig. 2 also shows (a) the average normalized reward gathered throughout single runs and (b) the average normalized LLM-Blender score yielded from 200 queries after the convergence of PickLLM to the

[3] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1
[4] https://huggingface.co/WizardLM/WizardLM-70B-V1.0
[5] https://huggingface.co/meta-llama/Llama-2-13b-chat-hf
[6] https://huggingface.co/meta-llama/Llama-2-70b-chat-hf
[7] https://huggingface.co/OpenAssistant/reward-model-deberta-v3-large-v2
[8] https://openai.com/blog/gpt-3-5-turbo-fine-tuning-and-api-updates
[9] https://huggingface.co/llm-blender/PairRM

Table 1. PickLLM vs alternatives: GPT3 and normalized LMM-Blender scores across different dataset topics

Sub Dataset from	PickLLM SLA		PickLLM QL $\epsilon = 0.1$		Mixtral		LLama2-70B	
HC3 [5]	GPT3	LLM-Blender	GPT3	LLM-Blender	GPT3	LLM-Blender	GPT3	LLM-Blender
reddit_eli5 [4]	0.79	0.61	0.80	0.58	**0.88**	**0.69**	0.87	0.59
open_qa [30]	0.95	0.51	0.93	**0.62**	0.95	0.50	**0.97**	0.49
wiki_csai [5]	**0.91**	0.60	0.93	**0.64**	0.90	0.63	0.90	0.56
medical [7]	**0.92**	**0.52**	0.90	0.49	0.89	0.43	0.91	0.51
finance [17]	0.85	0.46	0.88	0.44	0.88	**0.48**	**0.92**	0.43

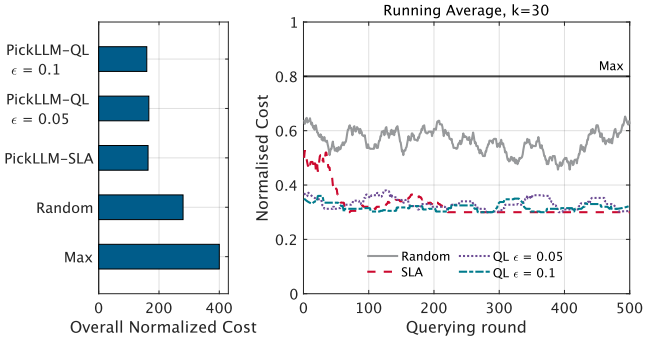

Figure 3. (Right) Overall cost, (Left) Cost per query

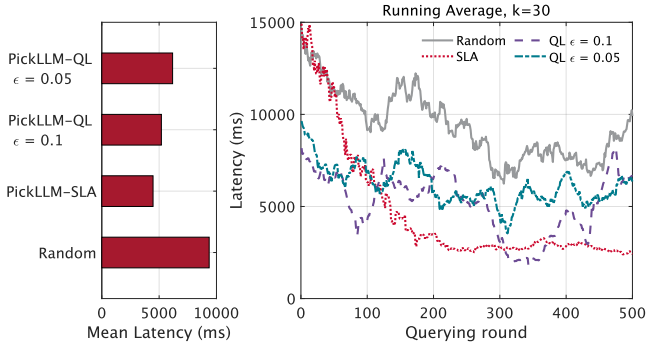

Figure 4. (Right) Mean Latency (ms), (Left) End-to-end latency per query round (ms)

optimal model. It is observed that as β increases, PickLLM converges to models that yield smaller rewards and produce responses of lower quality. Given the fact that PickLLM aims to optimize model choice in scenarios of thematically related small-batch user queries, we opt for an average learning rate that will balance convergence time and expected performance. Thus, for all the following experiments we set the SLA learning rate to $\beta = 0.5$, while the equivalent stateless Q-learning parameter θ is set to 0.7.

Next, we investigate the ability of PickLLM to adjust to specific optimization contexts specifically when users aim to optimize cost for their querying sessions, or when their underlying applications are time-critical requiring optimization concerning response latency. Thus, we examine how adjusting the weights of the proposed reward function achieves this goal. First, we fix the reward weights to $w_a = 0.2$, $w_c = 0.6$, $w_l = 0.2$, and run 500-query sessions from the WikiQA [30] portion of the HC3 dataset. Fig. 3-(Right) shows the overall normalized cost of each session for different variations of PickLLM, along with two corner cases of just using the most expensive choice throughout the session, or random model selection. PickLLM leads to lower total cost, while the SLA variation reduces the cost of the session 60% in comparison to the max, and less than 3% compared with the use of the Q-learning variations. This is attributed to the nature of the ϵ-greedy method that produces more exploration during the early portions of the session. Fig. 3-(Left) presents a running average of the real-time cost for each query round showcasing the convergence of PickLLM to a cost-efficient final state. Results are similar when the focus is set on optimizing latency for the same query set and reward function weights of $w_a = 0.2$, $w_c = 0.2$, $w_l = 0.6$. Fig. 4-(Right) shows the average latency of each session for the same variations of PickLLM, while Fig. 4-(Left) presents a running average of the real-time latency for each query round. Overall, in the best case scenario, mean latency is reduced by 52% over the random case, with the SLA variation showing less fluctuation over the alternatives.

Finally, we want to demonstrate the performance of PickLLM across datasets of different topics and do a comparison in terms of response performance. For the latter, two alternatives were considered: (a) using our largest model (parameter-size-wise), namely Llama2-70B for all queries, and (b) using a mixture of experts, namely Mixtral[10] [9] that has shown superior performance over many off-the-shelf LLMs. While the response quality of PickLLM can be as good as the underlying LLMs on the availability pool, our goal is to show that the outcome can be on par with state-of-the-art solutions while also allowing for adaption to optimization contexts set by the user. For PickLLM, we set reward function weights to $w_a = 0.33$, $w_c = 0.33$, $w_l = 0.33$, and all candidates perform 500-query sessions from each one of the sub-datasets of HC3. The responses of PickLLM and both

[10] https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

alternatives were also scored using the GPT3 judge and normalized LLM-Blender score mentioned above. Table 1 shows the final results of average GTP3 and LLM-Blender scores per session, that reflect the competitive performance of PickLLM across varying topics and contexts. The maximum scores for each sub-dataset and for each scoring method (GTP3 and LLM-Blender) are highlighted with bold.

5 Future Work

PickLLM aims for a simple way to guide agnostic users to an LLM choice that fits their data, and operational requirements. In the next steps, we consider the move from a model-free RL mechanism to a model-based RL approach that constructs a model of the environment to predict future states and rewards. For instance, model-based contextual multi-armed bandits, and statistical learners, such as neural networks, can be used to learn and generalize value functions to predict total returns (e.g., outcome from LLM choice) given a state and action pair. This will allow us to consider user context/conversation history towards steering users to models that are best suited for their type/style of questions. In addition, the reward function could be easily extended to include any factors of importance for the user e.g., GPU memory, environmental impact, licensing or other LLM usage constrains.

6 Conclusion

In this work, we underline the challenge of VCF customers to select the best-performing LLMs for their application given certain optimization contexts and resource constraints they may face. To tackle this issue, we propose PickLLM, a reinforcement-learning-based mechanism that examines the performance of the available LLMs while guiding queries to eventually the optimal model in terms of cost per query, overall latency, and response quality. Our approach is easily extendable to cover additional contextual optimization goals, while also being extremely computing-resource efficient in comparison to alternatives. Results indicate the adaptability of the proposed framework towards optimizing user querying sessions in terms of hard metrics with improvements of up to $\approx 50-60\%$ for overall session cost and average latency.

References

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

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* (2023).

[3] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv preprint arXiv:2305.05176* (2023).

[4] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. *arXiv preprint arXiv:1907.09190* (2019).

[5] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arXiv:2301.07597* (2023).

[6] Huanyu Guo, Donghua Yang, and Hong Gao. 2024. Reinforcement Learning-Based Adaptive Stateless Routing for Ambient Backscatter Wireless Sensor Networks. *IEEE Transactions on Communications* (2024).

[7] Xuehai He, Shu Chen, Zeqian Ju, Xiangyu Dong, Hongchao Fang, Sicheng Wang, Yue Yang, Jiaqi Zeng, Ruisi Zhang, Ruoyu Zhang, et al. 2020. Meddialog: Two large-scale medical dialogue datasets. *arXiv preprint arXiv:2004.03329* (2020).

[8] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).

[9] 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, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).

[10] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. *arXiv preprint arXiv:2306.02561* (2023).

[11] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems* 36 (2024).

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

[13] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110* (2022).

[14] Yixin Liu and Pengfei Liu. 2021. SimCLS: A simple framework for contrastive learning of abstractive summarization. *arXiv preprint arXiv:2106.01890* (2021).

[15] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. Routing to the Expert: Efficient Reward-guided Ensemble of Large Language Models. *arXiv preprint arXiv:2311.08692* (2023).

[16] Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. 2023. AutoMix: Automatically Mixing Language Models. *arXiv preprint arXiv:2310.12963* (2023).

[17] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*. 1941–1942.

[18] Kumpati S Narendra and Mandayam AL Thathachar. 2012. *Learning automata: an introduction*. Courier corporation.

[19] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. 2024. Routellm: Learning to route llms with preference data. *arXiv preprint arXiv:2406.18665* (2024).

[20] Mathieu Ravaut, Shafiq Joty, and Nancy F Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. *arXiv preprint arXiv:2203.06569* (2022).

[21] Marija Šakota, Maxime Peyrard, and Robert West. 2023. Fly-swat or cannon? cost-effective language model choice via meta-modeling.

arXiv preprint arXiv:2308.06077 (2023).

[22] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large Language Model Routing with Benchmark Datasets. *arXiv preprint arXiv:2309.15789* (2023).

[23] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári. 2000. Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine learning* 38 (2000), 287–308.

[24] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615* (2022).

[25] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[26] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171* (2022).

[27] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682* (2022).

[28] Francesc Wilhelmi, Boris Bellalta, Cristina Cano, and Anders Jonsson. 2017. Implications of decentralized Q-learning resource allocation in wireless networks. In *2017 ieee 28th annual international symposium on personal, indoor, and mobile radio communications (pimrc)*. IEEE, 1–5.

[29] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244* (2023).

[30] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2013–2018.

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

[32] Jieyu Zhang, Ranjay Krishna, Ahmed H Awadallah, and Chi Wang. 2023. EcoAssistant: Using LLM assistant more affordably and accurately. *arXiv preprint arXiv:2310.03046* (2023).

[33] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).

[34] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593* (2019).

A Prompt templates

We list the prompt template used for ChatGPT 3.5 Turbo when acting as an LLM judge [33]:

```
[System]
Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user question displayed below. We provide also a human generated response to use as guidance for your scoring. Rate the response on a scale of 0 to 1 with three decimal accuracy by strictly returning just one number, for example: "0.345".

[User Question]
question

[AI Response]
ai_response

[Human Response:]
human_response
```