MetaLLM: A High-performant and Cost-efficient Dynamic Framework for Wrapping LLMs

Quang H. Nguyen[1], **Thinh Dao**[1], **Duy C. Hoang**[1], **Juliette Decugis**[1],
Saurav Manchanda[2], **Nitesh V. Chawla**[3], **Khoa D. Doan**[1]
[1]College of Engineering and Computer Science, VinUniversity, Vietnam
[2]Amazon, USA*
[3]University of Notre Dame
{quang.nh, 21thinh.dd, duy.hc, juliette.d}@vinuni.edu.vn,
sauravm.kgp@gmail.com, nchawla@nd.edu, khoa.dd@vinuni.edu.vn

Abstract

The rapid progress in machine learning (ML) has brought forth many large language models (LLMs) that excel in various tasks and areas. These LLMs come with different abilities and costs in terms of computation or pricing. Since the demand for each query can vary, e.g., because of the queried domain or its complexity, defaulting to one LLM in an application is not usually the best choice, whether it is the biggest, priciest, or even the one with the best average test performance. Consequently, picking the right LLM that is both accurate and cost-effective for an application is necessary yet remains a challenge. In this paper, we introduce MetaLLM, a framework that dynamically and intelligently routes each query to the optimal LLM (among several available LLMs) for classification and multi-choice question-answering tasks, achieving significantly improved accuracy and cost-effectiveness. By framing the selection problem as a multi-armed bandit, MetaLLM balances prediction accuracy and cost efficiency under uncertainty. Our experiments, conducted on popular LLM platforms such as OpenAI and Together AI, as well as open-source LLM, showcase MetaLLM's efficacy in real-world scenarios, laying the groundwork for future extensions[1].

1 Introduction

Large language models have shown extraordinary zero-shot capabilities in various tasks and domains, such as text classification, summarization, question answering, and chatbot (Radford et al., 2019; Brown et al., 2020; Schick & Schütze, 2021; Wei et al., 2022; Ouyang et al., 2022; Trivedi et al., 2023). Recent works (Kaplan et al., 2020; Chowdhery et al., 2023; Hoffmann et al., 2022) suggest exhaustively scaling the model size and training data size to improve the performance of language models and provoke their emergent abilities; for example, GPT-4, with over 1.74 trillion parameters, achieves superior performance in several tasks but also incurs high economic costs. While this trend of scaling language models will continue in the future, there is also a growing diversification in recent models in terms of task or (sub-)domain specialization and computational costs. As a consequence, no single model –even the largest and most expensive – can consistently yield the best performance across all tasks. This means that, for model users, identifying which LLM is best suited for their applications will become crucial. However, this is a challenging task, especially when we factor in cost constraints, either in terms of computational resources or API service pricing.

*The work does not relate to the author's position at Amazon.
[1]Our code is available here.

We imagine a world with several LLM providers, such as OpenAI[2] or Together AI[3]; each provides service access to a diverse suite of LLMs with heterogeneous capabilities and cost structures. Here, an LLM user asks this important question: *How do I select an LLM i (out of k LLMs) for optimal performance and usage cost in my application?* One option is *combining multiple LLMs* as seen in existing ensemble methods (Jiang et al., 2023; Wang et al., 2023; Ong et al.), but this approach will yield significantly higher service costs; another option is cascading over a set of LLMs (Chen et al., 2023), but such an approach still requires querying the LLMs until we could find the best one. On the other hand, defaulting to a single LLM to avoid the extra cost of querying multiple models – by predicting the performance of LLMs (Shnitzer et al., 2023; Lu et al., 2023) – may not also be optimal; for some queries, a less expensive model may also provide correct answers. Furthermore, as different LLMs exhibit very distinctive abilities on different tasks and data distribution (Jiang et al., 2023; Wang et al., 2023), for a query, it is possible for LLM i to perform better than LLM j even though the cost of LLM i is noticeably lower than the cost of j.

In this work, we introduce a framework that wraps around a set of LLMs with diverse capabilities and brings the best performance at a more affordable cost to a user's application. The underlying component of this framework is a multi-arm bandit algorithm that routes each query to the least expensive LLM with the correct answer for the zero-shot classification and question answering task. Essentially, we formulate the problem of selecting the "best" LLM for a query as a decision-making problem under uncertainty: when a new query arrives, the proposed framework picks an LLM to answer and observe whether the LLM provides a correct answer; the goal is to maximize the total reward – a specific trade-off between the overall performance and usage cost. Intuitively, our method prefers the LLM that has a high probability of answering a query at a low cost. Our contributions can be summarized as follows:

- We propose MetaLLM, a versatile wrapper around a suite of any off-the-shelf LLMs, for zero-shot text classification tasks. MetaLLM can intelligently choose the target LLM for each query to achieve optimal performance and cost.
- We propose an algorithm based on multi-armed bandit to tackle the routing problem in MetaLLM. This algorithm is efficient since it makes the routing decision without needing to query any LLMs.
- Our experimental results on benchmark datasets and popular API services, including OpenAI and Together AI, and both closed-source and open-source LLMs, demonstrate the ability of MetaLLM in identifying the optimal LLM in terms of cost and performance. Specifically, MetaLLM improves the accuracy of the best model by around 1% while saving up to 60% and 10% of the total price on OpenAI and Together AI APIs, respectively.

Our work focuses on zero-shot classification and multiple-choice question-answering problems, which are common evaluation choices in the related work (Ong et al.). Nevertheless, the MetaLLM framework can be extended to arbitrary language tasks by modifying the reward function to incorporate suitable metrics assessing the quality of the responses. We leave it for future work.

2 Related Works

Large Language Models. The emergence of large language models (LLMs) has fundamentally transformed several domains, including natural language processing, computer vision, and e-commerce, and diverse tasks such as (zero-shot) classification, question-answering, and recommendation (Menghani, 2023; Liu et al., 2023a;b). The impressive effectiveness and versatility of these LLMs have come at the price of a drastic increase in LLM sizes, along with significant computational costs and data required to train, and expensive computational resources to perform inference with them. Consequently, several companies or services are now offering users access to LLMs with diverse capabilities and heterogeneous

[2] https://platform.openai.com/docs/models

[3] https://www.together.ai/

cost structures. For instance, the costs for processing 10 million tokens are \$0.1, \$0.18, \$0.2, and \$0.3 for Gemma, Llama, Mistral, and Qwen, respectively, using Together AI APIs. This abundance of API choices significantly burdens their users with the decision "*which LLM should I use?*", as different LLM APIs are known for their diverse capabilities for the prediction tasks (Liang et al., 2023; McKenzie et al.; 2022).

Prompt Optimization and Mixture of Experts. Fine-tuning is a standard option to improve the performance of LLMs for specific tasks. Mixture-of-Experts (MoE) (Eigen et al., 2013; Shazeer et al., 2017; Du et al., 2022; Si et al., 2023) trains a routing operator within the large model to enhance its performance; essentially, MoE assumes the model as a collection of "experts" (modules) and learn to route the input to the best expert. These approaches require training the LLMs, which is challenging for many users, while their single-LLM enhancements are usually model and scenario specific. Prompting reasoning like Chain-of-Thought (Wei et al., 2022; Wang et al., 2022; Zhou et al., 2022) or Tree of Thoughts (Yao et al., 2024) could improve the LLM performance without additional training. Both MoE and prompt-based reasoning, however, could not benefit from the large number of available LLMs, some of which could be significantly less expensive to use.

Model Ensemble. LLMs have been shown to yield diverse capabilities due to their architecture and dataset (Jiang et al., 2023; Wang et al., 2023). Jiang et al. (2023) observe that over 5000 instructions, the optimal LLM for each query significantly varies, and there is no single optimal model regardless of their size. Therefore they recommend ensembling to combine the strength of multiple LLMs for a better performance. FrugalML (Chen et al., 2020; 2022) cascades multiple machine learning models by querying sequentially until getting a response with a high confidence score. Motivated by FrugalML, Chen et al. (2023) exploit the full potential of LLMs by applying multiple techniques, including prompt engineering, caching, and cascading, for higher-quality answers.

Model Selection. Different from previous approaches that combine the strength of multiple models, there are some attempts to query a single LLM for each task. Hari & Thomson (2023) suggest training a language model to predict the performance of LLMs and route to the model with the highest performance, which incorporates the high cost of the router. Šakota et al. (2024) formulate the cost-performance trade-off as an integer linear programming (ILP) problem and use existing ILP solvers to assign appropriate LLMs for each input query. Lu et al. (2023) distill the preference of off-the-shelf reward models to select the LLM that achieves the best performance, ignoring its cost. Ding et al. (2024) train a router that maps easy queries to small models and hard queries to large models; however, their framework is only applicable when we have two LLMs. Recently, Hu et al. (2024) proposed a benchmark to evaluate routing models by their cost and performance on downstream tasks.

Commercialized Foundation Model Services. Recently, many companies have commercialized their LLM models so that customers can apply or even fine-tune LLMs for their own use cases without heavy technical knowledge. In 2023, OpenAI released many language models, such as text-ada-001, text-babbage-001, text-curie-001, and text-davinci-002[4], with various prices and capabilities. Their users can use these models as a chatbot, extract text embedding, or even ask them to write code. Recently, Together AI also brought forth their APIs that allow access to several foundation generative models, including text generation, image generation, and multimodal models, such as Llama, Gemma, Mistral, Qwen, Flux, etc.

3 MetaLLM Framework

3.1 Preliminaries

Zero-shot Classification. In this paper, we employ LLMs for text classifications without additional training. Given an input sentence $x \in \mathcal{X}$, we create a prompt to ask a language model M for a label. The model gives a correct prediction if the answer $M(x)$ matches the corresponding ground-truth label y.

[4]These models have been deprecated since 2024-01-04.

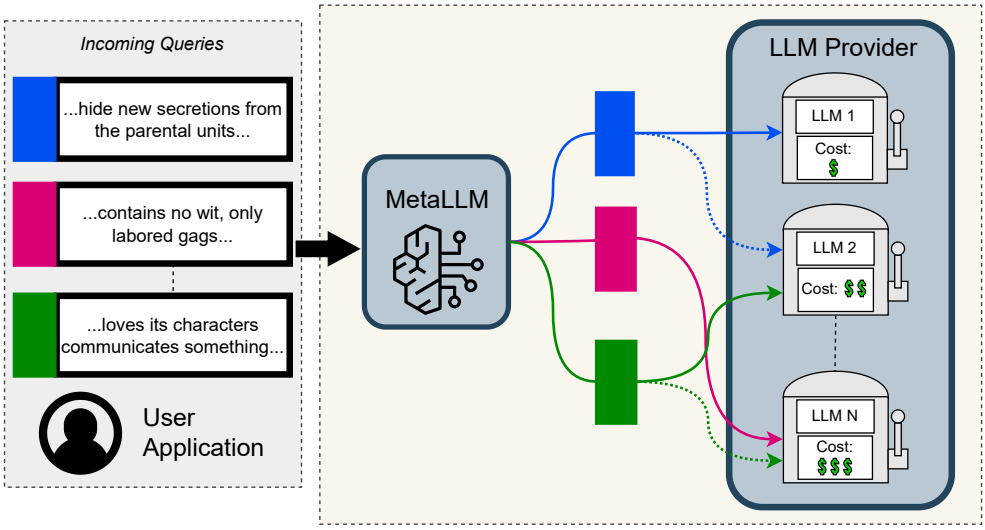

Figure 1: **The general process of serving queries in MetaLLM**. MetaLLM wraps around an existing LLM Provider, inspects each query, and then routes it to the least expensive LLM that can provide an accurate response. As an example, the **blue query** can be answered accurately by LLM 1 and LLM 2, but MetaLLM will route it to LLM 1 since it is less expensive; similarly in another example, the **green query** is routed to the least expensive LLM 2 even though LLMs 2 and 3 both can accurately answer it. The entire process is lightweight and can be performed without accessing in LLMs.

LLM APIs. We consider the case where the user has access to a set K of k different LLM APIs; each LLM M_i has a different capability, determined by how well M_i can answer a query x, and a cost c_i. For zero-shot classification, we represent the capability $a_i(x) \in \{0, 1\}$ of an LLM M_i on a sample x by comparing the answer to its corresponding ground-truth label: $a_i(x) = 1$ if $M_i(x) = y$. Usually, the model with a higher cost has better capability. If the user has a limited budget, they can choose less expensive models for their application. In contrast, if they require better performance, more expensive models will be more desirable. However, in general, the more expensive model is *not always* the best-performing choice for all queries (Jiang et al., 2023; Wang et al., 2023), making the LLM selection problem significantly challenging.

Problem Formulation. The goal of MetaLLM is to learn a routing function $f : \mathcal{X} \to K$ that dispatches a query x to an appropriate LLM to achieve a good response with a lower cost. For example, given a subset of LLMs, $K' \subseteq K$, that can give good responses for x, MetaLLM's objective is to return $\arg\min_{l \in K'} c_l$. In practice, the user wants the ability to balance the performance and the usage cost, depending on the needs of their application. More specifically, given a budget b, the user wants to maximize the performance on N queries while spending less than b to query the APIs.

$$\arg\max_{f} \sum_{i=1}^{N} a_{f(x_i)}(x_i) \quad \text{s.t.} \quad \sum_{i=1}^{N} c_{f(x_i)} \leq b \tag{1}$$

3.2 The optimal solution of routing objective

Let $S \in \{0, 1\}^{N \times k}$, $S_{i,j} = \mathbf{1}[f(x_i) = j]$, represent the choice of the routing function f on N queries, and $A \in \mathbb{R}^{N \times k}$, $A_{i,j} = a_j(x_i)$, represent the performance of the j-th API on each sample x_i. Eqn. (1) aims to choose the highest performance LLM within a cost constraint b

and can be reformulated as follows:

$$\arg\max_{S} \sum_{i=1}^{N}\sum_{j=1}^{k} A_{i,j}S_{i,j} \quad \text{s.t.} \quad \sum_{i=1}^{N} c_{f(x_i)} \leq b. \tag{2}$$

Instead of solving a discrete optimization problem, we relax $S \in \{0,1\}^{N\times k}$ to $S \in \mathbb{R}^{N\times k}$, $\sum_{j=1}^{k} S_{i,j} = 1$, and has the following dual problem (Boyd & Vandenberghe, 2004):

$$\arg\min_{p\in\mathbb{R}, q\in\mathbb{R}^N} p + \sum_i q_i \quad \text{s.t.} \quad pc_j + q_i \geq a_j(x_i) + 1. \tag{3}$$

Chen et al. (2022) study a similar formulation for model cascading and suggest solving the dual form as in Eqn. (3). In the case where we have an exact accuracy matrix A, we can optimize Eqn. (2) to find the optimal S and choose the suitable API j such that $S_{i,j} = 1$.

3.3 The Proposed MetaLLM

When deploying an application, we cannot know the exact accuracy of an LLM's API on a test sample before sending the test sample through this LLM. Previous works (Chen et al., 2022; 2020) learn a model to predict the performance of each API. These methods cascade multiple machine learning models and query them iteratively until the response has high confidence; consequently, they are highly expensive, especially when there are a substantial number of queries.

Algorithm 1 MetaLLM framework

input: k LLMs, each with cost c_j and accuracy $a_j(\cdot)$; cost scaling p, parameters θ, training sample $\{x_1, \ldots, x_n\}$, test query x.
output: The optimal LLM for the query x.

Initialize the policy
for j in 1..k **do**
  $A_j \leftarrow (X^\intercal X + \lambda I)$, $b_j \leftarrow X^\intercal y_j$
  $\theta_j^0 \leftarrow A_j^{-1} b_j$
end for

Inference
$j \leftarrow \arg\max_{j'} x^\intercal \theta_{j'} + \alpha\sqrt{x^\intercal A_{j'}^{-1} x}$
$A_j \leftarrow A_j + xx^\intercal$, $b_j \leftarrow b_j + r(x,j)x$
$\theta_j \leftarrow A_j^{-1} b_j$
return j

In this work, we approach this problem from a different perspective. Instead of training an accuracy predictor, we reformulate this problem as a multi-armed bandit. Specifically, for each input query, we define an LLM as an "arm", and obtain a reward expressing the performance of the LLM and the cost for that query. The benefit of this formulation is twofold: first, it allows the modeler to focus on designing the reward function to capture their application needs, making the framework more versatile; second, we can take advantage of the extensive and well-developed research on multi-arm bandits, including their rigorous theoretical foundations and practical solutions. The MetaLLM framework is depicted in Figure 1.

Reward function. The remaining question will discuss our design of the reward function that takes the input query and returns the reward of choosing the i-LLM. Chen et al. (2022) prove that if a cost scaling $p \in \mathbb{R}$ is the solution of Eqn. (3), the routing function $f(x) = \arg\max_i a_i(x) - pc_i$ will be the optimal solution of Eqn. (1). Intuitively, this strategy prefers the LLM with high performance and low-cost value. Motivated by that theoretical result, we propose the following reward function for training the multi-armed bandit:

$$r(x,i) = a_i(x) - pc_i. \tag{4}$$

Multi-arm bandit. We assume that the user has access to a training dataset of n queries and can obtain the training performance of each LLM on this dataset. We initialize the reward model $Q_j(x;\theta)$ of each arm j by minimizing the objective $\theta_j = \arg\min_\theta \|Q_j(x_i,\theta) - (a_j(x_i) - pc_j)\|_2^2$.

Similar to (Li et al., 2010), we employ a linear reward model $Q_j(x;\theta_j) = x^\intercal\theta_j$ and obtain the closed-form solution of ridge regression $\theta_j^0 = \left(A_j^0\right)^{-1} b_j^0, A_j^0 = (X^\intercal X + \lambda I), b_j^0 = X^\intercal y_j$, where X is the data matrix and $y_{j,i} = a_j(x_i) - pc_j$.

Given a new query x_t, we choose an arm using the UCB selection strategy:

$$j = \arg\max_{j'} x_t^\intercal \theta_{j'}^t + \alpha \sqrt{x_t^\intercal {A_{j'}^t}^{-1} x_t}, \quad \alpha > 0.$$

and update the reward function of the chosen arm, as follows:

$$\theta_j^{t+1} = \left(A_j^{t+1}\right)^{-1} b_j^{t+1}, A_j^{t+1} = A_j^t + x_t x_t^\intercal, b_j^{t+1} = b_j^t + r(x_t, j) x_t.$$

4 Experiments

In this section, we provide empirical results of MetaLLM on popular APIs and benchmark datasets.

4.1 Experimental Setup

LLM Services. We conduct our experiments with LLMs provided by popular API services, including OpenAI and Together AI. We chose four models from OpenAI: text-ada-001, text-babbage-001, text-curie-001, and text-davinci-002. These models have different costs and different capabilities, giving the users diverse options. The costs of these models are shown in Table 1.

Given a sample `SENT`, we query OpenAI models with the following prompt:

```
For the sentence: SENT, is the sentiment in this sentence positive or negative?
```

For Together AI APIs, we evaluate MetaLLM with four different LLMs: Gemma, Llama, Mistral, and Qwen. Since these models come from different families, this evaluation setting exhibits more heterogeneity than the setting with the OpenAI models. We summarize the cost of each APIs in Table 2.

Model	Cost
text-ada-001	\$0.40 / 1M tokens
text-babbage-001	\$0.50 / 1M tokens
text-curie-001	\$2.00 / 1M tokens
text-davinci-002	\$20.00 / 1M tokens

Table 1: Price of OpenAI APIs.

Model	Cost
Gemma	\$0.10 / 1M tokens
Llama	\$0.18 / 1M tokens
Mistral	\$0.20 / 1M tokens
Qwen	\$0.30 / 1M tokens

Table 2: Price of Together AI APIs.

Given a sample `SENT`, we query Together AI APIs with the following prompt:

```
Answer the following multiple-choice question with only the answer letter (A, B, C, or D
    ):
Question: QUESTION
Options:
A. CHOICE A
B. CHOICE B
C. CHOICE C
D. CHOICE D
Answer:
```

Datasets. We conduct experiments on SST-2 – a text classification dataset and MMLU – a question answering dataset. SST-2 is a binary sentiment analysis dataset consisting of movie reviews; the task here is to classify whether a review is positive or negative. This dataset has 67,439 training samples and 872 test samples. MMLU is a dataset multitask language understanding dataset including 57 multi-choice question answering subtasks. It has 14,042 test samples and 99,842 auxiliary training samples collected from several question answering benchmarks.

Setting	Method	Test cost	Test accuracy
	text-ada-001	0.096	80.50
	text-babbage-001	0.120	82.80
	text-curie-001	0.480	90.60
	text-davinci-002	4.800	91.51
Offline	MetaLLM (text-babbage-001 budget)	0.120	84.06
	MetaLLM (text-curie-001 budget)	0.539	89.56
	MetaLLM (text-davinci-002 budget)	2.030	91.97
Online w/o training data	MetaLLM (text-babbage-001 budget)	0.303	84.06
	MetaLLM (text-curie-001 budget)	0.818	87.27
	MetaLLM (text-davinci-002 budget)	1.456	88.30
Online	MetaLLM (text-babbage-001 budget)	0.117	84.06
	MetaLLM (text-curie-001 budget)	0.455	90.94
	MetaLLM (text-davinci-002 budget)	2.046	92.55

Table 3: The cost and accuracy of each LLM on OpenAI APIs

Training MetaLLM. For each input query, we utilize Sentence-BERT (Reimers & Gurevych, 2019) model to extract an embedding vector. MetaLLM is a linear model that maps the embedding vector to the reward expectation, which is optimized with the true reward (Eqn. (4)) by Algorithm 1. We normalize the cost of each LLM in the reward function such that the highest value is 1. For a budget b, we train MetaLLM with the scaling p five times, such that the cost of MetaLLM on the validation set is not higher than b, and compute the accuracy of the classification task. Given that p, we evaluate three different scenarios: i) only optimizing the reward function with training data, ii) only performing online updates during inference, and iii) initializing the policy from training data and updating during inference.

Evaluation. To evaluate, we compute the cost and the accuracy of MetaLLM on the test set of the classification task and compare them to each LLM candidate. We report the average cost per $10,000$ queries in all experiments.

4.2 Performance of MetaLLM on OpenAI Models

Table 3 shows the accuracy and cost of each OpenAI LLM and MetaLLM with different p on SST-2. As can be observed, the LLM with a higher cost has better accuracy. Noticeably, the differences in cost between LLMs can be very high; for example, text-davinci-002 is ten times more expensive than text-curie-001 but only has 0.4% better performance in terms of accuracy. Consequently, querying only the most expensive LLM is not an optimal choice unless the usage budget of the application is sufficiently high.

Performance of MetaLLM. For each LLM, except text-ada-001, we compute the cost of choosing that model only, fine-tune the scaling p on the validation set such that the cost is not higher than that model's cost (the budget), train MetaLLM with the found p five times and report the mean accuracy of the text classification task with LLMs selected by MetaLLM. We can observe that MetaLLM can achieve better accuracy at a lower cost. Specifically, with the same budget as text-babbage-001, MetaLLM's performance is 2% better than that of defaulting all queries to text-babbage-001, while having a slightly less expensive cost. Similarly, MetaLLM can achieve higher accuracy and lower cost than text-curie-001. More importantly, MetaLLM can reach a higher performance than that of defaulting to text-davinci-002, the best LLM, while spending 60% less.

Comparison of different scenarios. Table 3 also demonstrates that only optimizing on training data can yield higher accuracy than text-davinci-002, yet not as effective as online updating on test data. The initialization step also plays an important role; only online updating fails to find a good policy.

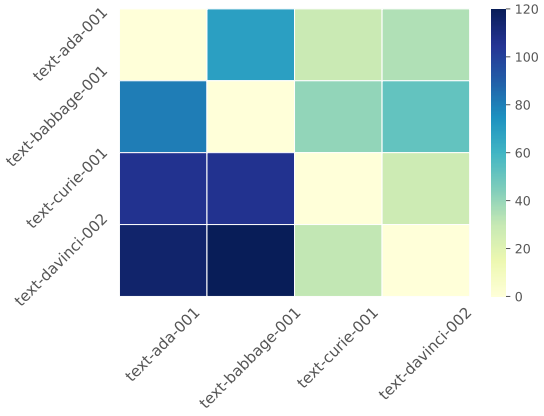

Figure 2: The number of samples that can be answered by one model but not by the other models. Cheaper models can answer many queries that more expensive models cannot.

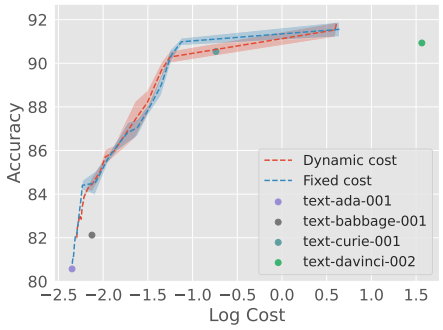

Figure 3: The cost-accuracy trade-off of MetaLLM with the cost in log scale for better visualization. MetaLLM with dynamic cost in the reward function slightly decreases the accuracy with a high budget; however, both approaches can perform better than a single LLM.

4.3 The Heterogeneous Capabilities of Different LLMs

The fact that MetaLLM can achieve better accuracy than even the most expensive OpenAI model means that an LLM's performance can vary across queries, and cannot always be determined by its usage cost (or the size of the LLM). In this section, we provide a more rigorous analysis of this observation. In Figure 2, the value at position (i, j) is the number of samples that are correctly classified by the i-th model but not the j-th model. As we can observe, there exist many queries that can be answered by the smaller models, such as text-ada-001 and text-babbage-001, while text-curie-001 and text-davinci-002 give incorrect answers. If MetaLLM can exploit this heterogeneous relationship between performance and cost to find the most suitable LLM for a query, it can even significantly boost the accuracy of using a single model. Our analysis in Section 4.6 confirms this hypothesis.

4.4 Reward Function With Dynamic Cost

The cost of an LLM is not fixed for every input; for the zero-shot classification task, it primarily depends on the length of the query. We perform an experiment where we use the exact cost of each training input in the reward function (4) and train MetaLLM with different p. The results for this dynamic cost setting and the fixed cost are provided in Figure 3. As we can observe, using dynamic costs does not lead to significant improvements, although both strategies can achieve better accuracies and lower costs than defaulting to a single LLM. When we set a high budget, dynamic cost setting can even has lower accuracy. We hypothesize that using dynamic cost imposes greater penalties on long queries, thereby encouraging MetaLLM to route them to cheaper models; these queries are, however, more complicated and may not be effectively solved by these cheaper models. Therefore, we recommend training MetaLLM with a fixed cost in the reward function for every training query.

4.5 Performance of MetaLLM on Together AI APIs

This section studies the scenario where the LLMs are heterogeneous, i.e., the less expensive models can perform better on some queries than the more expensive ones. We perform this experiment using the LLMs provided by Together AI and the MMLU dataset. Table 4 provides the accuracy and the cost of each approach, including defaulting to the same LLM (the first four rows) and our MetaLLM. As we can observe, when defaulting to a single LLM, a more expensive option does not guarantee a better performance; for example, Mistral is more expensive than Llama but yields lower performance.

Setting	Method	Test cost	Test accuracy
	Gemma	866.187	64.33
	Llama	1559.136	76.69
	Mistral	1732.374	66.68
	Qwen	2598.561	80.24
Offline	MetaLLM (Llama budget)	1474.975	75.70
	MetaLLM (Qwen budget)	2287.195	79.68
Online w/o training data	MetaLLM (Llama budget)	866.187	64.33
	MetaLLM (Qwen budget)	866.187	64.33
Online	MetaLLM (Llama budget)	1540.663	76.87
	MetaLLM (Qwen budget)	2304.500	80.90

Table 4: The cost and accuracy of each LLM on Together AI APIs

On the other hand, training MetaLLM with $p = 0$ yields the best performance (80.90% accuracy) on MMLU. Since the distribution of the auxiliary training set is far from the test set, optimizing the reward function does not provide a good policy, i.e., obtaining the highest accuracy. On the other hand, without the initialization step, the multi-arm bandit algorithm only defaults to Gemma. Setting a positive value to p decreases the cost substantially while having a minor degradation in the accuracy. Notably, MetaLLM can achieve better performance than the second-best LLM, Llama, but with a lower cost. These results further confirm that MetaLLM provides a cost-efficient yet high-performance policy.

4.6 Analysis of MetaLLM's Cost Scaling Parameter

In this section, we study the characteristics of MetaLLM with different values of p in the reward function. First, we consider the scenario where the user only optimizes for the performance (i.e., $p = 0$). Figure 4 shows the histogram or the frequency each OpenAI's LLM is selected by MetaLLM, separated by the number of correct and incorrect predictions, in the SST-2 dataset. As we can observe, MetaLLM indeed can learn whether an LLM can accurately answer a query. As discussed in Section 4.3, there exist many queries that can be correctly answered by less expensive models such as text-babbage-001, and MetaLLM chooses text-babbage-001 more often than choosing text-curie-001 or text-davinci-002 .

Figure 5 shows the histogram when putting a small regularization $p = 0.001$ on the cost component in the reward function. In this scenario, MetaLLM rarely picks text-davinci-002, while still being able to achieve 90.98% accuracy; as a reminder, this is still better than defaulting to the most expensive model, as previously observed in Table 3.

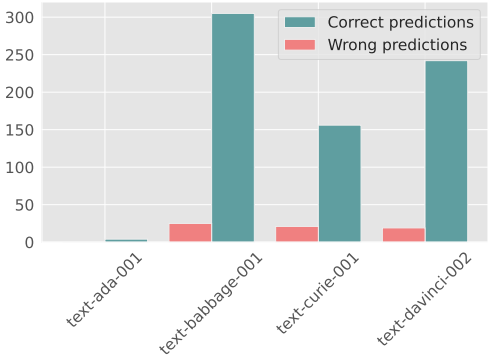

Figure 4: The histogram of OpenAI LLMs selected by MetaLLM with $p = 0$.

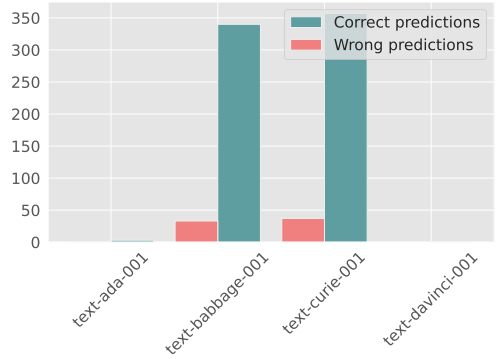

Figure 5: The histogram of OpenAI LLMs selected by MetaLLM with $p = 0.001$.

5 Conclusion

In this paper, we study the problem of dynamically selecting an LLM out of a set of LLMs for an input that achieves optimal performance and cost efficiency. To solve this problem, we propose a multi-armed bandit framework that can learn the reward of querying the correct model at a low price. Our approach, denoted as MetaLLM, is lightweight and applicable to any set of off-the-shelf LLMs and thus is versatile in practical use cases. Empirical results show that MetaLLM can improve the accuracy of the best API by around 1% while significantly reducing the cost of zero-shot text classification and multiple-choice question answering tasks by up to 60%.

6 Limitations and Societal Impacts

6.1 Limitations

As mentioned in the previous sections, we only study MetaLLM 's framework on zero-shot text classification tasks as these are important in NLP applications and increasingly utilize LLMs as the base predictors. Another reason is that it is straightforward to determine the correct LLM outputs and set up the reward function accordingly. As our paper aims to to demonstrate the potential of the MetaLLM's framework, this is sufficient.

However, the MetaLLM framework can be extended to arbitrary language tasks, such as as question answering or text generation, by modifying the reward function to incorporate suitable metrics assessing the quality of the responses. Due to the complexities of designing such reward function, these directions deserve independent studies. We leave them to future work.

MetaLLM also only trains a simple linear model whose input is the extracted feature of the query, which can ignore more fine-grained features. Building a more complex reward model and utilizing other information from the query, such as the domain of the input and the demand of the user, may further facilitate better the needs of the applications and improve the performance of MetaLLM.

Finally, we optimize MetaLLM with two values in the reward function: the performance and the cost of querying the API. However, several aspects to evaluate the model in practice could be incorporated into the reward, such as the inference time, the robustness of the model, emergent abilities, or even the information on the training distribution. Combining those factors can help build a more powerful and reliable AI system for diverse purposes.

6.2 Societal Impacts

Large language models, with their emergent abilities, have transformed our lives by assisting in several tasks. However, large models come with high inference costs; therefore they are very expensive to deploy and may cause harmful effects to the environment by high power consumption. Our framework helps reduce the cost of querying LLMs substantially by routing the input to cheaper models that may return the correct answer and can even improve performance by utilizing the combination of many LLMs. MetaLLM is applicable to any set of off-the-shelf LLMs, being useful for future AI systems with more modern language models.

References

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

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.

Lingjiao Chen, Matei Zaharia, and James Y Zou. Frugalml: How to use ml prediction apis more accurately and cheaply. *Advances in neural information processing systems*, 33: 10685–10696, 2020.

Lingjiao Chen, Matei Zaharia, and James Zou. Efficient online ml api selection for multi-label classification tasks. In *International conference on machine learning*, pp. 3716–3746. PMLR, 2022.

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*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language model. *arXiv preprint arXiv:2308.11601*, 2023.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

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.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023.

Peng Liu, Lemei Zhang, and Jon Atle Gulla. Pre-train, prompt, and recommendation: A comprehensive survey of language modeling paradigm adaptations in recommender systems. *Transactions of the Association for Computational Linguistics*, 11:1553–1571, 2023a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023b.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*, 2023.

Ian McKenzie, Alexander Lyzhov, Alicia Parrish, Ameya Prabhu, Aaron Mueller, Najoung Kim, Sam Bowman, and Ethan Perez. Inverse scaling prize: First round winners, 2022b. *URL https://irmckenzie. co. uk/round1*.

Ian McKenzie, Alexander Lyzhov, Alicia Parrish, Ameya Prabhu, Aaron Mueller, Najoung Kim, Sam Bowman, and Ethan Perez. Inverse scaling prize: Second round winners. *Fund for Alignment Research (FAR)*, 2022.

Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

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 from preference data. In *The Thirteenth International Conference on Learning Representations*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 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.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

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.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, 2021.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Tal Shnitzer, Anthony Ou, Mirian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. In *Annual Conference on Neural Information Processing Systems*, 2023.

Chenglei Si, Weijia Shi, Chen Zhao, Luke Zettlemoyer, and Jordan Boyd-Graber. Getting more out of mixture of language model reasoning experts. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 8234–8249, 2023.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10014–10037, 2023.

Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise. In *The Twelfth International Conference on Learning Representations*, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2022.