MixLLM: Dynamic Routing in Mixed Large Language Models

Xinyuan Wang[1]*, Yanchi Liu[2]†, Wei Cheng[2], Xujiang Zhao[2], Zhengzhang Chen[2], Wenchao Yu[2], Yanjie Fu[1], Haifeng Chen[2]

[1]Arizona State University, [2]NEC Labs America
{xwang735, yanjie.fu}@asu.edu
{yanchi, weicheng, xuzhao, zchen, wyu, haifeng}@nec-labs.com

Abstract

Large Language Models (LLMs) exhibit potential artificial generic intelligence recently, however, their usage is costly with high response latency. Given mixed LLMs with their own strengths and weaknesses, LLM routing aims to identify the most suitable model for each query in the stream to maximize response quality and minimize cost and latency. However, the challenges involve: (1) dynamic trade-offs among quality, cost, and latency; (2) enabling continual learning in deployed systems; and (3) navigating a varying (e.g., new LLM addition or old LLM removal) set of LLM candidates over time. To bridge these gaps, we develop MixLLM, a dynamic contextual-bandit-based routing system for query-LLM assignment. Specifically, we first leverage query tags to enhance query embeddings for the routing task. Next, we design lightweight prediction models to estimate the response qualities and costs of queries over LLMs. We then devise a meta-decision maker to choose the query-LLM assignments to best tradeoff response quality, cost, and latency. Finally, the system benefits from continual training, allowing it to adapt to evolving queries and user feedback over time. Our extensive experiments show that MixLLM achieves the best trade-offs in response quality, cost, and latency (97.25% of GPT-4's quality at 24.18% of the cost under the time constraint).

1 Introduction

Large Language Models (LLMs) have exhibited abilities to understand massive texts, generate actionable knowledge, enable contextual reasoning, and innovate diverse applications (Radford et al., 2018, 2019; Brown et al., 2020; Raffel et al., 2020; Chowdhery et al., 2023; Touvron et al., 2023). However, deploying LLMs presents unique challenges in managing computational resources, optimizing response times, and ensuring scalability.

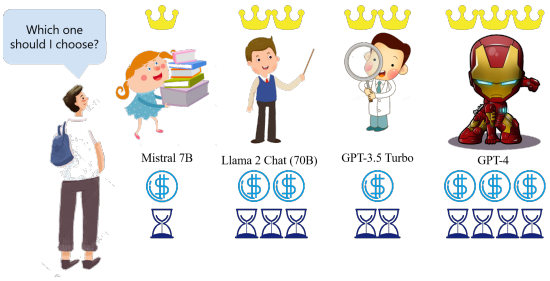


Figure 1: Which is the most suitable LLM?

As shown in **Figure 1**, the diversity of available LLMs (Jiang et al., 2024; Wang et al., 2022; Li et al., 2024; Wang et al., 2024b; Li et al., 2023; Wang et al., 2024a), each with different strengths and weaknesses, poses a challenge when selecting the most appropriate model for a given task. More powerful models, such as GPT-4 (Achiam et al., 2023), can deliver high-quality responses, but the pricy cost and computational requirements limit their accessibility. Thus, LLM routing, which chooses the most suitable LLMs for incoming queries in mixed LLM candidates, is needed to balance the trade-offs between response quality, cost, and latency.

Existing LLM routing methods can be categorized into non-predictive methods and predictive methods. Non-predictive methods, like cascading (Chen et al., 2023; Madaan et al., 2023), firstly exploit smaller LLMs and then switch to larger LLMs based on a reviewer model, but this increases both cost and latency as multiple LLMs are involved (Tay et al., 2022). Predictive methods predict the performance of candidate LLMs to select the best one for each query. For example, HybridLLM (Ding et al., 2024) exploits a binary classifier to predict the query difficulty for routing, RouterBench (Hu et al., 2024) predicts response quality directly, and FORC (Šakota et al., 2024) optimizes quality and cost at the set level.

However, the challenges involved in existing

*Work done during an internship at NEC Labs America.
†Corresponding author.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 10912–10922
April 29 - May 4, 2025

work are multifaceted. Firstly, a key limitation of current methods is the lack of consideration of high latency when too many queries are routed to the same LLM. Ignoring latency can create bottlenecks, reduce system efficiency, and impact user experience. Secondly, continual learning in a deployed system poses a significant challenge: LLMs must adapt to evolving queries and learn from user feedback over time to maintain relevance and accuracy, necessitating robust mechanisms for incorporating feedback. Lastly, flexibly managing the addition and removal of candidate LLMs is essential; as advancements in model architectures and techniques emerge, the routing system needs to integrate new models and retire outdated ones to ensure users benefit from the latest advancements.

To address these challenges, we propose MixLLM, a dynamic contextual-bandit-based routing system for query-LLM assignment. First, we propose a tag-enhanced embedding model by using tags generated from the InsTag (Lu et al., 2023) model. These tags help improve the query representations from noises. Next, we design lightweight prediction models for each LLM to estimate response quality and cost. These LLM-specific predictions do not require system-wide retraining when new LLMs are introduced. The meta decision-maker then selects the best LLM for each query based on the predictions. It balances trade-offs between response quality, cost, and latency to optimize query-LLM assignments. Finally, MixLLM benefits from continual training, allowing the system adapts to evolving queries and user feedback over time, improving the performance in real-world deployment.

Our extensive experiments demonstrate that MixLLM effectively balances response quality, cost, and latency, achieving 97.25% of GPT-4's quality at only 24.18% of the cost. By incorporating a latency penalty, MixLLM avoids congestion and high-latency issues, ensuring efficient system performance even under heavy load. Additionally, we extend the RouterBench dataset by incorporating the latest Llama 3.1 model, showcasing the framework's scalability and adaptability. The results from online training further validate the effectiveness of the continual training approach.

Our contributions are as follows:

- We propose MixLLM that leverages enhanced query embeddings, latency penalties, and continual learning to balance response quality, cost, and latency in LLM routing.
- MixLLM accounts for real-world query streams by introducing a latency mechanism that factors in hardware limitations.
- MixLLM offers key benefits, including selecting the optimal LLM, handling the latency constraint, and adapting over time to changing environments and user feedback.
- We extend the RouterBench dataset by incorporating the latest Llama 3.1 model and adding prompt and response length.

2 Related Work

The studies of selecting the most suitable LLM can be categorized into *non-predictive* and *predictive* routing systems.

2.1 Non-predictive Routing System

Non-predictive systems incorporate LLM inference during routing. A common approach is *cascading*, where smaller models are used first, switching to larger ones if needed. FrugalGPT (Chen et al., 2023) introduced three strategies to reduce cost while maintaining response quality: prompt adaptation, LLM approximation, and LLM cascade form a chain of LLMs, selecting LLMs from small to large. AutoMix (Madaan et al., 2023) introduced a similar cascading strategy, where a self-reviewer judges the answer and a meta-reviewer decides whether switching to a larger model is needed. However, in non-predictive routing systems, one query may need to be answered by several LLMs which increases both cost and resource usage.

2.2 Predictive Routing System

Predictive systems estimate the quality of LLM response before making routing decisions and route each query to only one LLM. These systems typically fall into categories such as classification, quality prediction, optimization, and bandit-based solutions, each offering unique strategies.

Classification-based approaches predict the best LLM for a query by treating LLMs as labels. HybridLLM (Ding et al., 2024) trained a binary classifier to assign "easy" queries to smaller models. ME-Switch (Liu et al., 2024b) extended to a multi-label domain classifier, improving memory and computation efficiency. Other methods like Zooter (Lu et al., 2024) introduced a reward model for ranking responses from different LLMs, using tag-based label enhancement for training data. RouteLLM (Ong et al., 2024) introduced four distinct routing strategies, including similarity-weighted ranking, matrix

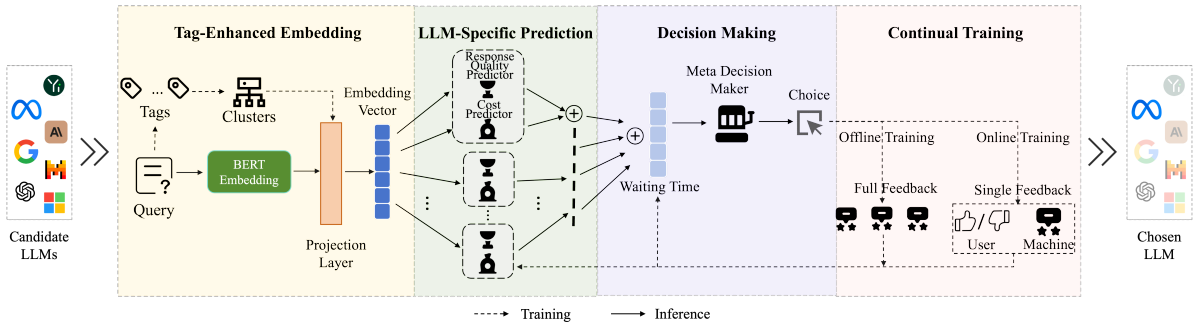


Figure 2: Overview of the MixLLM Framework

factorization, and supervised and prompting classification. However, query labels may shift when new and powerful LLMs emerge.

Response quality prediction methods focus on predicting the quality of each LLM's response for a given query. Shnitzer et al. (Shnitzer et al., 2023) used 3 different ways to predict "correctness" (response quality) for each LLM and select the best one. RouterBench (Hu et al., 2024) optimized the quality-cost trade-off by a "willingness to pay" parameter. It also introduced a large benchmark dataset for routing tasks. However, they did not predict cost and ignored latency.

Optimization-based methods treat LLM routing as a set-level optimization problem. FORC (Šakota et al., 2024) employed predicted response quality and cost for quality-oriented and cost-oriented linear programming strategies. OptLLM (Liu et al., 2024c) optimized the routing problem with a multi-label classification model. But these approaches potentially ignore low cost-effective queries.

Bandit-based methods like MetaLLM (Nguyen et al., 2024) adopted a single bandit approach, where the system learns to balance quality and cost trade-offs over time. However, the dependency between arms can limit scalability when adding or removing LLMs.

3 Methodology

3.1 The Dynamic LLM Routing Task

We study the problem of dynamic LLM routing with streaming queries. Given queries that arrive sequentially, our goal is to assign each query to the most appropriate LLM selected from a set of candidates to trade off response quality, cost, and latency. Formally, let the set of streaming queries be: $Q = \{q_n\}_{n=1}^{|Q|}$, and the set of LLM candidates be: $M = \{m_l\}_{l=1}^{|M|}$. The objective is to select the most suitable LLM m_n^* for the query q_n.

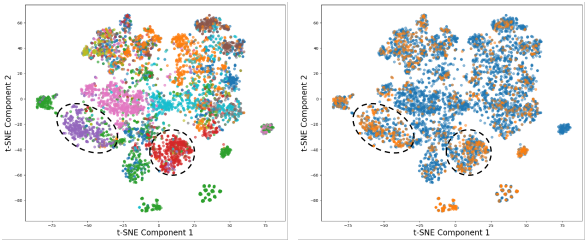
3.2 Overview of The MixLLM Framework

Figure 2 shows that MixLLM consists of four components: (1) tag-enhanced query embedding, (2) LLM-specific prediction, (3) meta decision maker, and (4) continual learning mechanism. This framework allows MixLLM to route queries to LLMs in a dynamic system while achieving quality-cost-latency trade-offs and continual learning with a changing LLM candidate set.

3.3 Tag-enhanced Query Embedding via Unsupervised Fine-tuning

A query can be seen as a token sequence, thus, its embedding can be generated using a pre-trained encoder (e.g., BERT (Devlin et al., 2019)): $\mathbf{e}_n = \texttt{Encoder}(q_n)$, where $\mathbf{e}_n$ represents the embedding of n-th query q_n in a query stream. However, such general-purpose query embeddings contain too much noises and are not tailored for LLM routing. To address this limitation, we propose enhancing the encoder by introducing tag knowledge, which enriches the query embeddings and improves their effectiveness for routing tasks.

Different LLMs can be proficient in different domains (e.g., Science, Legal) (Liu et al., 2024a). Using GPT-4 as an example, **Figure 3** shows a clear correlation between domain and response quality. The query distribution after t-SNE dimension reduction is shown in **Figure 3a**, with each color representing a specific domain. **Figure 3b** highlights GPT-4's response quality. It is evident that GPT-4 has a higher error frequency (orange points in **Figure 3b**) in the "Legal" (red points in **Figure 3a**) and "Math" (purple points in **Figure 3a**) domains. These observations inspire us to develop the tag-enhanced embedding approach. By incorporating tags and their derived domains, we can guide embeddings to capture these distinctions, making them more suitable for LLM routing tasks.



(a) Domain Visualization (b) Quality Visualization

Figure 3: Domain-Quality Correlation

Step 1: Automated Query Tag Generation. To prepare, we employ the InsTag (Lu et al., 2023) model to generate fine-grained tags for each query and manually cluster the tags into a set of coarse-grained domains, denoted as D. InsTag is an instruction tagging method designed to quantify the diversity and complexity of human instructions, and these tags contribute to model fine-tuning.

Step 2: Unsupervised Fine-tuning of Encoder. While the InsTag model, backed by Llama-2 13B, is too large to be used during inference, we fine-tune the BERT encoder during the training stage. We develop an unsupervised optimization objective that integrates intra-domain similarity ($\mathcal{L}_{\texttt{intra}}$) and inter-domain separation ($\mathcal{L}_{\texttt{inter}}$):

$$\mathcal{L} = \mathcal{L}_{\texttt{intra}} + \mathcal{L}_{\texttt{inter}}, \tag{1}$$

where the intra-domain similarity loss encourages embeddings within the same domain cluster to be close to their center $\boldsymbol{\mu}_j$:

$$\mathcal{L}_{\texttt{intra}} = -\frac{1}{|Q|}\sum_{i=1}^{|Q|}\log\frac{\exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_i)}{\sum_{j=1}^{|D|}\exp(\mathbf{e}_i \cdot \boldsymbol{\mu}_j)}. \tag{2}$$

The inter-domain separation loss ensures that different domain centers are distinct:

$$\mathcal{L}_{\texttt{inter}} = \frac{1}{|D|}\sum_{j=1}^{|D|}\log\sum_{k\neq j}\exp(\boldsymbol{\mu}_j \cdot \boldsymbol{\mu}_k). \tag{3}$$

3.4 LLM-Specific Quality and Cost Prediction

Given a query embedding, we aim to predict both the response quality and financial cost for each candidate LLM on the query, so the meta decision-maker can assign the most suitable model.

Step 1: Estimating the Response Quality of A Query-LLM Pair. Since different LLMs have different response qualities, we learn an LLM-specific regression function for each LLM. This function estimates the response quality of the n-th query on the l-th LLM:

$$\hat{p}_{n,l} = f_l^{\texttt{rq}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\texttt{rq}}), \tag{4}$$

Step 2: Estimating the Financial Cost of A Query-LLM Pair. The total cost of the n-th query on the l-th LLM includes: 1) the known input cost and 2) the predicted output cost, according to typical LLM pricing policies:

$$\hat{c}_{n,l} = \underbrace{\texttt{len}_{n,l}^{\texttt{prm}} \cdot \texttt{price}_l^{\texttt{prm}}}_{\text{input cost}} + \underbrace{\hat{\texttt{len}}_{n,l}^{\texttt{res}} \cdot \texttt{price}_l^{\texttt{res}}}_{\text{output cost}}, \tag{5}$$

where $\texttt{len}_{n,l}^{\texttt{prm}}$ is the prompt length of query q_n, and $\texttt{price}_l^{\texttt{prm}}$ and $\texttt{price}_l^{\texttt{res}}$ are unit prices of input prompt and output response. The response length $\hat{\texttt{len}}_{n,l}^{\texttt{res}}$ is predicted using a similar method as the response quality predictors:

$$\hat{\texttt{len}}_{n,l}^{\texttt{res}} = f_l^{\texttt{rl}}(\mathbf{e}_n; \boldsymbol{\theta}_l^{\texttt{rl}}), \tag{6}$$

3.5 Meta Decision Maker

For the n-th query q_n, the final decision score for each candidate LLM is determined by three factors: (1) $s_{n,l}^{\texttt{trade}}$, which trade-offs the predicted quality and cost; (2) $s_{n,l}^{\texttt{unc}}$, which accounts for potential prediction uncertainty; and (3) $s_l^{\texttt{pen}}$, which discourages selecting candidates with long waiting time:

$$s_{n,l} = s_{n,l}^{\texttt{trade}} + \alpha \cdot s_{n,l}^{\texttt{unc}} - \beta \cdot s_l^{\texttt{pen}}. \tag{7}$$

where α and β control the relative importance.

The willingness to pay λ is introduced in $s_{n,l}^{\texttt{trade}}$ to control the priority of quality over cost, leading to different budgets accordingly:

$$s_{n,l}^{\texttt{trade}} = \frac{\lambda}{\lambda+1} \cdot \hat{p}_{n,l} - \frac{1}{\lambda+1} \cdot \hat{c}_{n,l}, \tag{8}$$

To handle prediction errors, we introduce an uncertainty measurement ($s_{n,l}^{\texttt{unc}}$) to enhance robustness (Li et al., 2010):

$$s_{n,l}^{\texttt{unc}} = \mathbf{e}_n^T \cdot \mathbf{A}_l^{-1} \cdot \mathbf{e}_n, \tag{9}$$

where $\mathbf{A}_l^{-1}$ represents the inverse covariance matrix for the l-th LLM. This measures the amount of information gathered for each candidate and adjusts the confidence of the prediction accordingly.

Considering hardware limitations, it is crucial to avoid routing queries to candidates with excessively long waiting times. The penalty is therefore given by:

$$s_l^{\texttt{pen}} = e^{\gamma \cdot (w_l - \xi \cdot \tau)}, \tag{10}$$

where γ is a scaling factor and τ represents the maximum tolerable waiting time. The waiting time w_l for candidate l includes: 1) the initial latency required for the LLMs to start and 2) the token output time for generating each token in the response. The coefficient ξ (smaller than 1) makes the penalty stronger. By scaling the threshold to $\xi \cdot \tau$, the system applies the penalty earlier, discouraging the selection of candidates before their waiting time reaches the full limit of τ.

Finally, the candidate with the highest score among all candidates is selected as the most suitable one:

$$m_n^* = \arg\max_l(s_{n,l}) \tag{11}$$

3.6 Continual Learning

To ensure effectiveness in real-world applications, we designed both offline and online training modes. The offline mode enables the model to achieve robust performance before deployment, while the online mode allows the model to continuously improve in response to changing environments and user feedback.

Offline Training: Prior to deployment, we perform offline training using refined full feedback from all candidate LLMs. The refined feedback includes real response quality and length, which involves updating the parameters of the predictive models:

The parameters $\boldsymbol{\theta}_l^{\mathsf{rq}}$ for the response quality predictors are updated using gradient descent:

$$\boldsymbol{\theta}_l^{\mathsf{rq}} := \boldsymbol{\theta}_l^{\mathsf{rq}} - \eta_1 \cdot \nabla_{\boldsymbol{\theta}_l^{\mathsf{rq}}} \mathcal{L}(p_{n,l}, \hat{p}_{n,l}), \tag{12}$$

Similarly, the response length predictor parameters $\boldsymbol{\theta}_l^{\mathsf{RL}}$ are updated as:

$$\boldsymbol{\theta}_l^{\mathsf{rl}} := \boldsymbol{\theta}_l^{\mathsf{rl}} - \eta_2 \cdot \nabla_{\boldsymbol{\theta}_l^{\mathsf{rl}}} \mathcal{L}(\mathtt{len}_{n,l}^{\mathsf{res}}, \hat{\mathtt{len}}_{n,l}^{\mathsf{res}}), \tag{13}$$

The uncertainty matrices $\mathbf{A}_l$ are updated incrementally by query embeddings:

$$\mathbf{A}_l := \mathbf{A}_l + \mathbf{e}_n^T \cdot \mathbf{e}_n. \tag{14}$$

This update accumulates information over time, decreasing the inverse $\mathbf{A}_l^{-1}$, which leads to low uncertainty, indicating increased confidence in predictions. Then the waiting time is adjusted based on the LLM assignment.

Online Training: After deployment, the system incrementally updates predictive models and uncertainty matrices using refined single feedback from the selected LLMs.

However, user feedback based on human satisfaction with the LLM service, often binary ("good" or "not good"), is challenging for training. To address this, we introduce a Dynamic Feedback Score ($s_{n,l}^{\mathsf{df}}$) based on the contextual bandit method to capture the binary user feedback and dynamically adjust the scoring mechanism.

The final score for each LLM is updated as:

$$s'_{n,l} = s_{n,l} + \kappa_{n,l} \cdot s_{n,l}^{\mathsf{df}}, \tag{15}$$

where $s_{n,l}^{\mathsf{df}}$ represents the appropriateness of the l-th LLM to answer the given query predicted by a shared 3-layer MLP network:

$$\left[s_{n,1}^{\mathsf{df}}, s_{n,2}^{\mathsf{df}}, \ldots, s_{n,|M|}^{\mathsf{df}}\right] = f^{\mathsf{df}}(\mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}). \tag{16}$$

And $\kappa_{n,l}$ is the confidence factor based on the variance, to ensure the reliability of $s_{n,l}^{\mathsf{df}}$ and prevent over-reliance on unstable predictions:

$$\kappa_{n,l} = \frac{1}{\mathrm{Var}_n[s_{n,l}^{\mathsf{df}}] + \epsilon}, \tag{17}$$

where ϵ is a small constant to avoid division by zero. Low variance increases $\kappa_{n,l}$, which will enhance the importance, while high variance decreases it, which reflects instability. Then the candidate with the highest score is selected:

$$m_n^* = \arg\max_l(s'_{n,l}) \tag{18}$$

Since we cannot directly supervise the network outputs with binary feedback r_n, we apply the **Policy Gradient** method (Ban et al., 2021) to update $\boldsymbol{\theta}^{\mathsf{df}}$. The probability of selecting candidate l is:

$$\pi(l \mid \mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}) = \frac{\exp\left(s_{n,l}^{\mathsf{df}}\right)}{\sum_{k=1}^{|M|} \exp\left(s_{n,k}^{\mathsf{df}}\right)}. \tag{19}$$

The goal is to maximize the expected reward:

$$J(\boldsymbol{\theta}^{\mathsf{df}}) = \mathbb{E}_{l \sim \pi(\cdot|\mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}})}[r_n], \tag{20}$$

with gradient on selected candidate m_n^*:

$$\nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \log \pi(m_n^* \mid \mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}) = \nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \left(s_{n,m_n^*}^{\mathsf{df}} - \log \sum_{k=1}^{L} \exp\left(s_{n,k}^{\mathsf{df}}\right) \right) \tag{21}$$

The parameters are updated as:

$$\boldsymbol{\theta}^{\mathsf{df}} := \boldsymbol{\theta}^{\mathsf{df}} - \eta_3 \cdot \nabla_{\boldsymbol{\theta}^{\mathsf{df}}} \log \pi(m_n^* \mid \mathbf{e}_n; \boldsymbol{\theta}^{\mathsf{df}}) \cdot r_n. \tag{22}$$

4 Experiments

4.1 Experimental Settings

4.1.1 Dataset

We conduct our experiments utilizing the RouterBench dataset (Hu et al., 2024), which consists of 36,497 queries from 8 NLP datasets, including Chinese and English. Each query is answered by 11 different LLMs, with records of responses, as well as corresponding quality and cost metrics. Moreover, we extend the dataset with Llama 3.1 8B and 70B models [1] and add prompt and response lengths of all the queries and responses. The dataset is split into 80% training and 20% testing.

4.1.2 Baseline Algorithms

We compare MixLLM with both non-predictive and predictive baselines in our experiments. For non-predictive methods, the cascading approach tests smaller models first and switches to larger ones if needed. We extend AutoMix (Madaan et al., 2023) by ordering multiple LLMs by size, with cheaper models prioritized when sizes are equal. For predictive methods, RouteLLM (Ong et al., 2024) assigns queries to LLMs using a BERT-based multi-label classifier, while Zooter (Lu et al., 2024) is represented by an MLP-based classifier. RouterBench (Hu et al., 2024) predicts response quality to achieve a quality-cost trade-off. Both FORC (Šakota et al., 2024) and OptLLM (Liu et al., 2024c) predict quality and then perform set-level optimization, while MetaLLM (Nguyen et al., 2024) uses a bandit algorithm with a quality-cost reward. For additional comparison, we also include random routing and individual LLMs.

Since the baseline algorithms do not include online training after deployment, we only compare them with our offline training component for a fair comparison in Section 4.2, while the online training component is further evaluated in Section 4.3.

4.1.3 Evaluation Metrics

We evaluate the methods on the streaming test queries based on the quality-cost trade-off under the latency constraint. Specifically, the response quality score for each query is scaled from 0 to 1, while the query cost is measured in dollars. Any query that exceeds the maximum tolerable waiting time is assigned a quality score of 0. The total quality and total cost are calculated as the sum of quality scores and query costs for all the test queries. We evaluate the routing performance across varying budget levels using parameter λ, ranging from 10^{-6} to 10^{6} in Equation (8), with a larger λ will prioritize response quality.

4.1.4 Configurations

In our experiments, we set up a software environment consisting of Python 3.12, PyTorch 2.0, and CUDA 12.1 running on Ubuntu 18.04 LTS. Most experiments were conducted on a 12GB Titan-V GPU, while tasks involving Llama models, such as dataset extension and tag extraction, were performed on two 80GB H100 GPUs.

All random seeds are set to 42 for reproducibility. In Equation (7), α is set to 0.01, and β is set to 0.1. In Equation (10), γ is set to 0.1.

As for learning rates, η_1 and η_2 are set to 1, reflecting the use of simple machine learning algorithms, while η_3 is set to 0.001 due to the complexity of the neural network.

Query streams are configured at a rate of 100 queries per 10 seconds. The maximum tolerable waiting time τ is set to 30 seconds, and the waiting time of LLMs will be updated every 10 seconds. The prices of input and output, the average initial time, and response speeds of different LLMs are publicly available [2]. This website estimates the costs of open-source LLMs based on computational resources, including CPU, GPU, and memory usage, while API-based LLMs are priced directly using their API rates.

As for quality and length regressors, we use random forest (RF) for quality prediction across all LLMs, while a combination of multi-layer perceptron (MLP), RF, and K-nearest neighbors (KNN) is applied for length (cost) prediction depending on the LLM. Those predictors are lightweight. For example, the size of an MLP model is less than 2MB, so the inference and update time is shorter.

4.2 Overall Results

As shown in **Figure 4**, MixLLM consistently outperforms the baselines, delivering strong performance. For the baseline methods, response quality can decline with larger budgets since queries may exceed the latency constraint. Notably, MixLLM achieves 97.25% of GPT-4's quality at only 24.18% of the cost when λ is 1.4. In comparison, the best baseline method, OptLLM, reaches 96.39% of GPT-4's quality at 32.94% of the cost. However, beyond this point, OptLLM's response quality drops

[1] https://ai.meta.com/blog/meta-llama-3-1/

[2] https://artificialanalysis.ai/

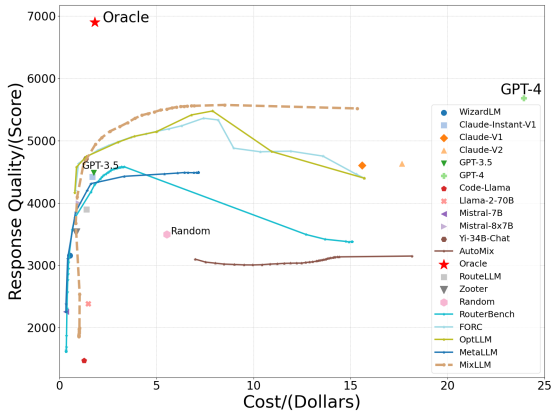


Figure 4: Overall Results

as many queries exceed the waiting time tolerance, while MixLLM remains stable. The same situation also happens on other baseline algorithms.

The *Oracle* result represents the most optimal routing on this dataset, balancing response quality and cost. It serves as a benchmark for the best possible assignment. In this context, a point closer to the upper left (*Oracle*) signifies higher quality at a lower cost. To obtain the *Oracle* result, all candidate LLMs are tested for each query. For each query, the LLM that meets the quality threshold and has the lowest cost is selected. While the final results reflect only the quality and cost of the selected LLM, the process of determining the *Oracle* result requires significant computational resources.

Each single LLM provides one quality-cost point. For instance, GPT-4 demonstrates superior quality, while GPT-3.5 offers a better balance of cost and quality. The "Random" routing serves as a baseline; points above and to the left of this anchor are superior in offering better quality at a lower cost.

AutoMix struggles because multiple LLMs handle each query, quickly exhausting the budget and reaching the latency constraint. RouteLLM and Zooter fail to adjust budgets dynamically and can only provide one quality-cost point. RouterBench performs well at lower budgets but faces latency issues as budgets increase. FORC and OptLLM share the problem of ignoring some queries due to set-level optimization, affecting user experience. MetaLLM is less effective because it can't consider multiple LLMs simultaneously, underscoring the need for a multi-armed bandit approach.

4.3 Study on Continual Training

To enable continual training, we simulate the real-world query streams by splitting the training dataset into different ratios (**Table 1**) for offline and online training. For example, an 80:20 split means 80% of the data are used in offline training, while 20% of the data are used in online training. The offline training uses refined feedback across these splits. For online training, in addition to the refined feedback, user feedback is simulated by assuming the user is satisfied if the response quality exceeds 0.7 and the waiting time is less than 15 seconds.

Table 1 presents the overall response quality for each setting, calculated as the sum of the response qualities divided by the total number of queries. Higher percentages indicate better performance. To ensure fairness, results within the same split ratio (column) maintain similar costs, which means the improvements reflect the impact of online training feedback. Results show both types of feedback improve model performance. Although the improvement may seem modest, it's important to note that online feedback is only available for the selected one, which limits the effectiveness compared to offline training. Despite this limitation, the results suggest that online training becomes more effective as more data are available. In real-world scenarios, where online training data are abundant, MixLLM will have greater opportunities to adapt. Refined feedback outperforms binary feedback due to its detailed nature. Nevertheless, even the simpler binary feedback contributes to improved performance.

Table 1: The Power of Continual Training

Setting	**Offline : Online**		
	80:20	**50:50**	**30:70**
Without Online Training	75.54%	71.98%	69.74%
With Refined Feedback	76.45%	72.99%	71.29%
Improvement	**1.21%**	**1.39%**	**2.22%**
With Binary Feedback	75.93%	72.37%	70.65%
Improvement	**0.52%**	**0.53%**	**1.31%**

In our experiments, we implemented one online test at the end of online training to demonstrate the continuing improvement of learning from and aligning with online feedback. Without loss of generality, we believe our one-time finding (online feedback can improve performance and alignment) can be generalized to recurrent tests. It is feasible to adapt our system to conduct recurrent tests at the end of each cycle in a real-world scenario.

4.4 Study on Tag-Enhanced Embedding

To obtain tags for the tag-enhanced encoder training, we employ InsTag (Lu et al., 2023), a Llama-

based tagging LLM to generate one or more tags for each training query. InsTag is capable of producing over 6,000 tags, e.g., "data structure", "legal ethics", which are manually categorized into 20 domains, e.g., "Computer Science", "Legal".

Table 2: Effect of Tag-Enhanced Embedding

Cost Level	Cost Range	General Embedding	Enhanced Embedding	Improvement
Low	< 1	53.14%	56.18%	**5.72**%
Middle	1 - 8	72.09%	73.43%	**1.85**%
High	> 8	75.76%	76.36%	**0.79**%

The results in **Table 2** demonstrate the effectiveness of tag-enhanced embedding. The values represent response qualities across different cost levels, where each cost level corresponds to a specific cost range. To ensure fairness, the costs within each level (row) are kept similar. As the cost level increases, which corresponds to a higher budget and a greater emphasis on response quality, the improvement from tag-enhanced embedding diminishes. Nevertheless, at each cost level, tag-enhanced embedding consistently enhances routing performance, highlighting its importance.

4.5 Study on Latency Constraint

Theoretically, the trade-off between response quality and query cost often operates within the bounds of limited hardware resources in the real world. Effectively managing the workload on devices becomes essential. Different components, such as the CPU, GPU, memory, and bandwidth, all have their performance metrics, but these factors converge on one critical metric: query waiting time. Therefore, we employ the latency as the primary constraint.

We conducted a simulation to account for the latency constraint. The total time required to answer a query has two parts: 1) the initial time to begin generating and 2) the response time, which depends on the answer length. We use the average initial time for each LLM and estimate the response time by multiplying the output length by the corresponding LLM's generation speed. For closed-source LLMs, the simulation is based on API statistics. For open-source LLMs, we simulated under ideal hardware conditions, assuming sufficient memory and stable network connections to ensure optimal performance. The average initial time and response speed of different LLMs are publicly available [3].

[3] https://artificialanalysis.ai/

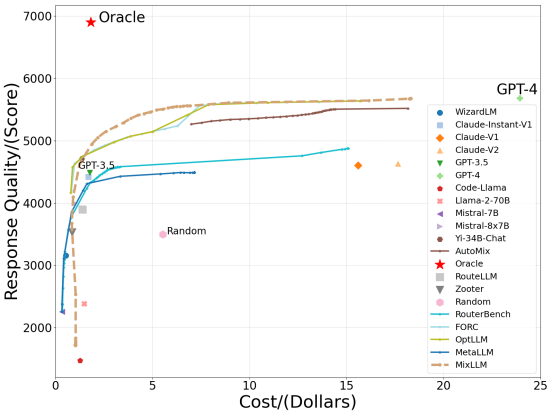


Figure 5: Results without Latency Constraint

Even without the latency constraint, MixLLM still outperforms the baselines, as shown in **Figure 5**. When compared to results with latency constraint (**Figure 4**), MixLLM maintains stable performance due to the time penalty component. However, the baselines show more variation.

In **Figure 4**, AutoMix's performance drops the most, primarily due to its cascading nature. Each query starts with the first LLM, resulting in significantly increased waiting time. Other predictive baselines also experience performance declines at higher cost levels, as they tend to route queries to more powerful LLMs with longer waiting times. This results in many queries exceeding the maximum tolerable waiting time and going unanswered.

4.6 Study on Adaptive Training

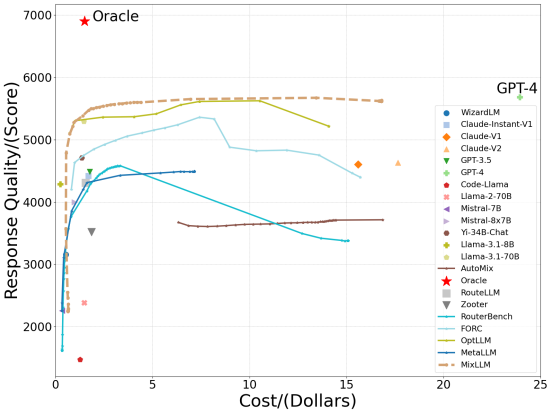


Figure 6: Results with Updated Candidates

Each LLM in MixLLM operates independently, ensuring scalability. Adding or removing candidate LLMs does not require complete re-training, which only affects the corresponding LLM. To demonstrate this advantage, we extended the RouterBench

dataset using new Llama 3.1 models. Specifically, we utilized the Llama 3.1 8B and 70B models to answer each query in the dataset. Then we record responses and measure their quality, cost, and length. As shown in **Figure 6**, with the introduction of the powerful Llama 3.1 models, MixLLM achieves 98.55% of GPT-4's response quality while reducing the cost to just 16.79% when λ is 1.8. Furthermore, MixLLM continues to outperform other baselines.

4.7 Study on Out-of-Domain Generalization

Real-world queries often originate from new or unseen domains, presenting challenges for LLM routing systems. To evaluate the domain adaptation and generalization capabilities of MixLLM, we conducted an out-of-domain (OOD) experiment. In this setup, we simulate an OOD scenario using the domains defined by tags. We maintain an 80:20 splitting ratio, where the testing set (20% of the data) contains non-overlapping domains not present in the training set (80% of the data). This design ensures that some testing samples belong to entirely unseen domains during training.

Table 3: Result on OOD Scenario

Splitting Policy	Offline Only	Offline + Online
Normal 80:20 Splitting	75.54%	76.45%
OOD 80:20 Splitting	71.43%	73.89%
Decrease	5.44%	**3.35%**

The results in **Table 3** reveal that when using only offline training, MixLLM's performance decreases by 5.44% at the same price cost level. However, when integrating both offline and online training, the performance drop is mitigated to 3.35%. This demonstrates that the integrated offline-online training strategy effectively enhances domain generalization and adaptation. Furthermore, we identify MixLLM's OOD problem as a novel routing task, calling on the research community to explore and incorporate advanced domain adaptation techniques into frameworks like ours to better address this pressing challenge.

4.8 Study on Different Choice Policy

During our experiments, a new question arises: *Can selecting more LLMs improve performance?* To explore this, we applied various selection policies, with the results presented in **Figure 7**.

"Top 1", "Top 2", and "Top 3" refer to policies where the LLM(s) with the highest 1, 2, or 3 scores are selected. When multiple LLMs are chosen, the response quality reflects the best one, while costs are summed. The "TOP 1.5" policy introduces a dynamic adjustment, which selects the top 1 LLM when the budget is low and expands to include more LLMs as the budget increases. As illustrated in **Figure 7**, increasing the number of selected LLMs shifts the curve upwards and to the right. This outcome is expected because selecting more LLMs increases both cost and the likelihood of choosing the most capable model. Notably, with the same budget (red line in **Figure 7**), the "Top 3" policy achieves the highest response quality, even surpassing the most powerful single LLM, GPT-4, at only 20% of its cost.

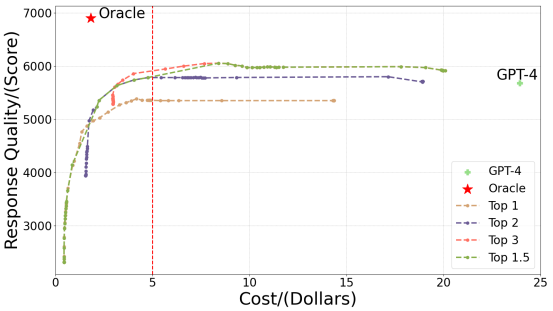


Figure 7: Results on Choice Policy Study

However, in practical scenarios, users typically seek a single, definitive answer rather than multiple options. *How to select the final answer?* Adding a reviewer to choose the best answer is one potential solution, but it requires additional time and resources. Given the complexity, we did not incorporate a multi-choice selection into MixLLM. It presents interesting engineering challenges, and we welcome further exploration and collaboration for those interested in addressing this problem.

5 Conclusion

We proposed MixLLM, a dynamic routing system that selects the most suitable LLM for each query by balancing response quality, cost, and latency. By enhancing query embeddings with tag knowledge and incorporating latency constraints, MixLLM effectively addresses key challenges in real-world LLM deployment. The system's adaptability, achieved through continual learning and independent prediction for each LLM, ensures efficiency as queries evolve and new models are introduced. Our results demonstrate that MixLLM optimizes resource usage while maintaining strong performance across varying budget levels.

Limitations

Although MixLLM presents strong performance in the experiments, some limitations are listed as follows. (1) The training process assumes access to refined feedback, including response quality and cost, which may not always be available in real world. Training-free methods could help, such as scaling laws (Ruan et al., 2024). (2) MixLLM may face challenges when routing queries from brand-new domains, commonly referred to as the out-of-domain (OOD) problem (see Section 4.7 for further details). (3) MixLLM faces challenges in practical scenarios requiring the selection of a single definitive answer from multiple LLM outputs, as discussed in Section 4.8. (4) While MixLLM considers hardware limitation through the latency constraint, more detailed dispatch strategies considering system information could further improve its practicality. (5) More complex routing tasks remain unexplored, such as hierarchical routing. This could involve first routing a query to a relevant domain, and then selecting the most suitable LLM within that domain. (6) MixLLM's performance needs to be tested in real-world applications to ensure its robustness beyond idealized environments.

Acknowledgments

Dr. Yanjie Fu is supported by the National Science Foundation (NSF) via the grant numbers: 2426340, 2416727, 2421864, 2421865, 2421803, and National Academy of Engineering Grainger Foundation Frontiers of Engineering Grants.

References

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*.

Yikun Ban, Jingrui He, and Curtiss B Cook. 2021. Multi-facet contextual bandits: A neural network perspective. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 35–45.

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

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*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. *arXiv preprint arXiv:2404.14618*.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. 2024. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*.

Yushan Jiang, Zijie Pan, Xikun Zhang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. 2024. Empowering time series analysis with large language models: A survey. *arXiv preprint arXiv:2402.03182*.

Haozhou Li, Qinke Peng, Xinyuan Wang, Xu Mou, and Yonghao Wang. 2023. Sehf: A summary-enhanced hierarchical framework for financial report sentiment analysis. *IEEE Transactions on Computational Social Systems*.

Haozhou Li, Xinyuan Wang, Hongkai Du, Wentong Sun, and Qinke Peng. 2024. Sade: A speaker-aware dual encoding model based on diagbert for medical triage and pre-diagnosis. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12712–12716. IEEE.

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

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024a. MathBench: Evaluating the theory and application proficiency of LLMs with a hierarchical mathematics benchmark. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6884–6915, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Jing Liu, Ruihao Gong, Mingyang Zhang, Yefei He, Jianfei Cai, and Bohan Zhuang. 2024b. Meswitch: A memory-efficient expert switching framework for large language models. *arXiv preprint arXiv:2406.09041*.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. 2024c. Optllm: Optimal assignment of queries to large language models. *arXiv preprint arXiv:2405.15130*.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2024. Routing to the expert: Efficient reward-guided ensemble of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1964–1974, Mexico City, Mexico. Association for Computational Linguistics.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

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*.

Quang H Nguyen, Duy C Hoang, Juliette Decugis, Saurav Manchanda, Nitesh V Chawla, and Khoa D Doan. 2024. Metallm: A high-performant and cost-efficient dynamic framework for wrapping llms. *arXiv preprint arXiv:2407.10834*.

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*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Yangjun Ruan, Chris J Maddison, and Tatsunori Hashimoto. 2024. Observational scaling laws and the predictability of language model performance. *arXiv preprint arXiv:2405.10938*.

Marija Šakota, Maxime Peyrard, and Robert West. 2024. 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*, pages 606–615.

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*.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, et al. 2022. Ul2: Unifying language learning paradigms. *arXiv preprint arXiv:2205.05131*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Xinyuan Wang, Haozhou Li, Dingfang Zheng, and Qinke Peng. 2024a. Lcmdc: Large-scale chinese medical dialogue corpora for automatic triage and medical consultation. *arXiv preprint arXiv:2410.03521*.

Xinyuan Wang, Qinke Peng, Xu Mou, Haozhou Li, and Ying Wang. 2022. A hierarchal bert structure for native speaker writing detection. In *2022 China Automation Congress (CAC)*, pages 3705–3710. IEEE.

Xinyuan Wang, Liang Wu, Liangjie Hong, Hao Liu, and Yanjie Fu. 2024b. Llm-enhanced user-item interactions: Leveraging edge information for optimized recommendations. *arXiv preprint arXiv:2402.09617*.