Adaptive LLM Routing Under Budget Constraints

Pranoy Panda[*] **Raghav Magazine**[*†] **Chaitanya Devaguptapu**
Sho Takemori **Vishal Sharma**[‡]
Fujitsu Research
pranoy.panda@fujitsu.com, email@chaitanya.one, takemori.sho@fujitsu.com

Abstract

Large Language Models (LLMs) have revolutionized natural language processing, but their varying capabilities and costs pose challenges in practical applications. LLM routing addresses this by dynamically selecting the most suitable LLM for each query/task. Previous approaches treat this as a supervised learning problem, assuming complete knowledge of optimal query-LLM pairings. However, real-world scenarios lack such comprehensive mappings and face evolving user queries. We thus propose to study LLM routing as a contextual bandit problem, enabling adaptive decision-making using bandit feedback without requiring exhaustive inference across all LLMs for all queries (in contrast to supervised routing). To address this problem, we develop a shared embedding space for queries and LLMs, where query and LLM embeddings are aligned to reflect their affinity. This space is initially learned from offline human preference data and refined through online bandit feedback. We instantiate this idea through ***P**reference-prior **I**nformed **L**inUCB f**O**r Adaptive Rou**T**ing* (PILOT), a novel extension of LinUCB. To handle diverse user budgets for model routing, we introduce an online cost policy modeled as a multi-choice knapsack problem, ensuring resource-efficient routing.

1 Introduction

Deploying Large Language Models (LLMs) in real-world systems faces a critical challenge: balancing performance with cost-effectiveness (Li et al., 2024). While larger models offer superior performance, their high costs makes their universal deployment impractical. This challenge is particularly acute given the varying pricing structures of proprietary models and the resource requirements of deploying the open-source alternatives.

To understand the need for varying resource requirements, consider a customer service chatbot handling diverse queries. For simple queries like "*What are your business hours?*", a smaller, cost-effective model might suffice. However, for complex inquiries, such as "*I'm torn between two of your smartphone models: the X200 and the Z300. I need a phone with excellent battery life, a high-quality camera, and robust performance for multitasking. Can you provide a detailed comparison, including any potential drawbacks of each model?*", a more powerful (and costly) model may be necessary to ensure better planning and reasoning capabilities, which a smaller model might lack. This scenario illustrated the need for dynamic query routing – The ability to dynamically route queries to the most appropriate model based on complexity and cost considerations, known as the *model-routing* problem (Ding et al., 2024).

Existing approaches model the routing problem as a supervised learning task, requiring large-scale labeled datasets mapping queries to their optimal LLM pairings (Ding et al., 2024; Hu et al., 2024). This problem formulation faces two limitations: (1) Gathering such labeled datasets is very expensive, as it requires responses from each model in the model pool for every query to discover the optimal query-LLM pairing. (2) Lack of adaptability to change in query distribution.

Thus, to find a more practical problem setting, we draw parallels with news recommendation systems, where models can only learn from user feedback, such as clicks on a single article (Li et al., 2010; Bouneffouf and Rish, 2019). These systems must predict the best article for a user without showing all possible articles , receiving feedback solely on the selected article. Similarly, LLM routing in-

[*]Equal contribution

[†]Current affiliation: Microsoft Research; The author contributed while working at Fujitsu Research of India; email: raghavmagazine@gmail.com

[‡]Current affiliation: Microsoft; The author contributed while working at Fujitsu Research of India; email: i.vishal1990@gmail.com

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 23934–23949

volves choosing the best model for a query, where obtaining user feedback on all models is labor-intensive and costly.

Building on this insight, *we reformulate LLM routing as a contextual bandit learning problem* - a formulation commonly used in news and ad recommendation scenarios (Bouneffouf and Rish, 2019). Thus, instead of requiring outputs from every LLM to identify the best match, our problem setting relies only on a binary bandit feedback, i.e., whether the chosen LLM's response is good or not. This approach is practical, as simple feedback mechanisms, like thumbs up/down ratings (i.e. like/dislike feedback), are now common in chat interfaces (Appcues, 2024; Delighted, 2024), allowing effective learning from user interactions without need for extensive annotation across LLMs.

To address this newly formulated problem of LLM routing with bandit feedback, we propose to develop an evolving shared embedding space for queries and LLMs, where distances represent routing affinity. Initially pretrained on human preference data (Chiang et al., 2024), it is refined through online user feedback. Furthermore, to enforce cost constraints in an online setting, we introduce a novel policy modeled as an online multi-choice knapsack problem (Chakrabarty et al., 2008), dynamically allocating resources to balance budget adherence and performance. While authors in (Nguyen et al., 2024) have approached the routing problem from a bandit perspective, their work was focused on the limited setting of classification tasks and did not incorporate explicit online cost constraints.

Our key contributions are as follows,
(i) We formulate LLM routing as a budget constrained contextual bandit problem for adaptive decision-making with limited supervision
(ii) We propose a Preference-Prior Informed LinUCB algorithm (PILOT) that combines offline human preference data with online bandit feedback to route queries (Sec 2.2.2). We also show our preference prior helps achieve a lower regret bound than standard algorithm. This algorithm is further coupled with an online cost policy to dynamically allocate cost budget to queries (Sec 2.2.3).
(iii) In Sec 4 we show that our method outperforms existing bandit baselines across datasets, achieving lowest regrets and highest performance across different cost budgets.

2 Methodology

2.1 Problem Formulation

As discussed earlier, this work tackles the problem of routing queries to Large Language Models (LLMs) in an online setting, learning solely from evaluative (bandit) feedback from users. Leveraging these user interactions, our goal is to maximize overall performance under budget constraints, effectively personalizing LLM selection over time. We now present the formal problem statement.
Let $L = \{l_1, l_2, \ldots, l_k\}$ be a set of k LLMs, $\mathcal{Q}$ be the space of all natural language queries, and $\mathcal{Y}$ be the space of all natural language responses. A query $q_t \in \mathcal{Q}$ at time t is represented by its embedding $x_t \in \mathbb{R}^{d_e}$ in a d_e-dimensional space, generated by a pre-trained embedding model $\phi : \mathcal{Q} \to \mathbb{R}^{d_e}$. We assume black-box access to each LLM $l_i \in L$, where the response of LLM l_i to query q_t is denoted as $y_t^{l_i} \in \mathcal{Y}$. The quality of a response is quantified by a scoring function $s : \mathcal{Q} \times \mathcal{Y} \to [0, 1]$, derived from human feedback or heuristic-based metrics.
At each time step $t = 1, \ldots, Q$, for a given query $q_t \in \mathcal{Q}$, an LLM router $M : \mathcal{Q} \to L$ selects an LLM $l \in L$. After receiving the response from the selected LLM l, the router observes a reward $r_t = s(q_t, y_t^l) \in [0, 1]$ representing the quality of the response. Each LLM $l_i \in L$ also incurs a token cost $C_t^{l_i} \geq 0$ for processing query q_t. The objective of the LLM router is to maximize the total reward, defined as $\sum_{t=1}^{Q} r_t$, while satisfying the budget constraint $\sum_{t=1}^{Q} C_t^{M(q_t)} \leq B$, i.e for Q consecutive queries, it ensures that the total token cost across these queries is less than B.

2.2 Proposed Method

We will now describe our solution approach to LLM routing that hinges on learning an effective mapping from queries to the LLMs. For this, we will learn (i) an embedding of a given query and (ii) an embedding for each LLM in a shared embedding space such that the cosine distance between a query and an LLM represents their mutual affinity. The query-LLM shared embedding space is not static; rather, it evolves through online training.
While the online bandit feedback may be sufficient to train such an embedding space, it may still take a considerable amount of time to train in a completely online fashion. Further, a vast amount of public data is available in the form of human preferences where given a query and responses from

two LLMs, humans provide their preferred LLM response. We hypothesize that independent of the end task (model routing in our case), this human preference data can be leveraged to pretrain the shared embedding space. The online bandit feedback can then be used at run time to continuously improve the pretrained embedding space, thus enhancing the accuracy of our routing decisions over time. Hence, we leverage two primary sources of information, (i) offline human preference data (Section 2.2.1) to pretrain the shared embedding space and (ii) online bandit feedback (Section 2.2.2) to continuously tune the embedding space at runtime. Finally, to address the critical aspect of user-level budget constraints, we implement an online cost policy (Section 2.2.3). However, its worth noting that in the online bandit learning phase (Section 2.2.2), there is no budget constraint. - we elaborate on this in Section 3. This multi-faceted approach allows us to balance performance optimization with budget adherence in a dynamic, user-centric manner. We now describe these steps in detail. For a birds eye view of the method, see Algorithm 1 & 2.

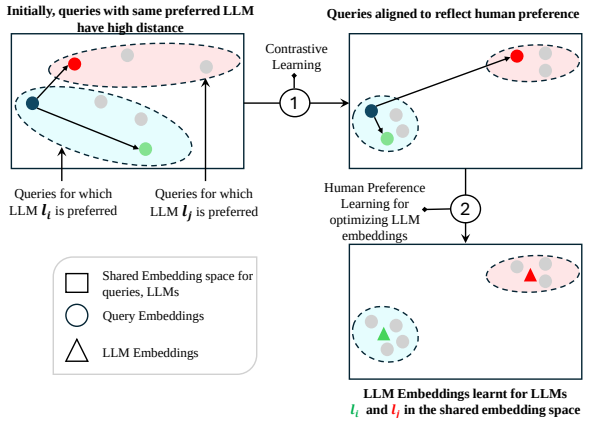


Figure 1: **Pretraining with Human Preference Data** (1) We leverage human preference dataset to learn query embeddings which are aligned w.r.t. human preferences on query-LLM mapping. Then, in (2) we learn LLM embeddings aligned with projected queries

2.2.1 Pretraining with Human Preferences

Human preference data provides rich insights into query-LLM fit. We use it to establish a meaningful shared embedding space before incorporating online bandit feedback on the target task. Pretraining occurs in two phases for stability, mitigating the moving target problem that can arise from jointly optimizing query projections and LLM embeddings via cosine similarity. In phase one, we learn a projection from (initial) query embeddings onto our shared embedding space. In phase two, using same human preferences, we learn an embedding for each LLM that lies in the shared embedding space.

Phase one: Learning the Query Projections Given an existing query embedding model $\phi : \mathcal{Q} \rightarrow \mathbb{R}^{d_e}$, we project these embeddings to our d_m-dimensional shared space via a learned linear transformation $\psi(q) = W\phi(q) + b$. The parameters $W \in \mathbb{R}^{d_m \times d_e}$ and $b \in \mathbb{R}^{d_m}$ are learned using a cosine distance-based triplet loss on human preference data D_{pref} (Figure 1 (1)). For each anchor query $(q_a, l_i, l_j, l_{win}) \in D_{\text{pref}}$ (where l_{win} is the preferred LLM), we construct positive and negative query pools. The positive pool $P = \{(q, l_i, l_j, w) \in D_{\text{pref}} | \ \ l_w = l_{win}\}$ contains queries where l_{win} was also preferred. The negative pool $N = \{(q, l_i, l_j, w) \in D_{\text{pref}} | \ \ l_w \neq l_{win} \wedge size(l_w) < size(l_{win})\}$ consists of queries where l_{win} was not preferred against a smaller LLM (hard negatives, based on token cost - size(l)).

Phase two: Learning LLM Embeddings In phase two, we focus on learning the LLM embeddings θ_i for each LLM $l_i \in L$. For this, we first freeze the query projection parameters (W and b) learned in phase one and then learn the LLM embeddings with the goal that given a query $(q, l_i, l_j, l_w) \in D_{\text{pref}}$, the embedding of the preferred LLM (l_w) is close to $\psi(q)$. We start by defining a probability distribution of l_i winning over l_j as $p_i = \frac{\exp(\cos(\theta_i, \psi(q)))}{\sum_{k \in \{i,j\}} \exp(\cos(\theta_k, \psi(q)))}$. Then, we train the LLM embeddings by treating preference learning as a binary classification task using binary cross-entropy loss. The final learned embeddings after this phase for an LLM l_i is denoted as θ_i^{pref} (See (2) in Figure 1).

This two-phase learning process establishes an initial shared embedding space that captures the relationship between queries and LLMs based on human preferences, providing a strong foundation for subsequent online learning.

2.2.2 Evolving with Online Bandit Feedback

Having learned query and LLM embeddings, we will now discuss how to incorporate the online bandit feedback for the end task of routing queries to appropriate LLMs. For this, we model routing as a contextual multi-armed bandit (CMAB) problem: projected query embeddings $\psi(q_t)$ serve as contexts, and LLMs are arms. Reward $r_t = s(q_t, y_t^l) \in [0, 1]$ is the response quality from the selected LLM l. Objective is to maximize cumulative reward

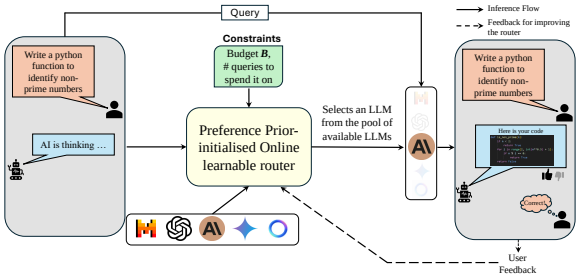


Figure 2: **Bandit Router Framework:** Our router takes three inputs: (i) User query (ii) cost constraints and, (iii) a model pool. It learns and adapts automatically based on user feedback, optimizing LLM selection over time.

$\max_\pi \mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$ via policy $\pi : \mathbb{R}^{d_m} \to \mathcal{A}$.

The bandit's feedback is used to further align query and LLM embeddings with the goal of routing the query to appropriate LLM. For this, we choose to update arm/LLM embeddings at each round (following existing literature).

Let θ_a^t denote embedding of arm (LLM) a at time step t where the initial arm embedding at time $t = 0$ is initialized with LLM embeddings learned in pretraining phase from preferences, i.e., $\theta_a^0 = \theta_a^{\text{pref}}$. As discussed in Section 2.2.1, we model the mutual affinity between user queries and LLMs as the cosine distance between their representations in the shared embedding space. In line with this idea, we model the expected reward at time t in the CMAB problem as follows:

$$\mathbb{E}[r_t|a, q_t] = \cos(\hat{\psi}(q_t), \hat{\theta}_a) = \hat{\psi}(q_t) \cdot \hat{\theta}_a \quad (1)$$

where $\hat{\psi}(q_t) = \frac{\psi(q_t)}{\|\psi(q_t)\|_2}$ and $\hat{\theta}_a = \frac{\theta_a}{\|\theta_a\|_2}$. Thus, owing to this linear reward formulation (cosine distance between unit normalized vectors), we propose a preference-prior informed linear upper confidence algorithm (PILOT) which builds upon the standard LinUCB (Li et al., 2010) algorithm while incorporating the knowledge gained from our preference learning phase.

PILOT, similar to LinUCB, performs online ridge regression. From a Bayesian perspective, this corresponds to maintaining a posterior distribution over the arm parameters. At each time step t, for each arm a, the point estimate of embedding of the arm (representing LLM a) is given by:

$$\tilde{\theta}_a^t = (A_a^t)^{-1} b_a^t \quad (2)$$

where, $A_a^t = A_a^{t-1} + \hat{\psi}(q_t)\hat{\psi}(q_t)^\top$, and, $b_a^t = b_a^{t-1} + r_t\hat{\psi}(q_t)$. At $t = 0$, we initialize the parameters A_a^t and b_a^t as, $A_a^0 = \lambda_a I$, $b_a^0 = \lambda_a \theta_a^{\text{pref}}$, where θ_a^{pref} is the embedding for LLM corresponding to arm a learned from preference data (see Section 2.2.1), and $\lambda_a > 0$ is a regularization parameter. This initialization can be interpreted as imposing a human preference prior $\hat{\theta}_a^0 \sim \mathcal{N}(\theta_a^{\text{pref}}, (\lambda_a I)^{-1})$ on the (initial) arm parameters. λ_a controls prior strength: larger λ_a implies less exploration (lower variance), while smaller λ_a allows faster adaptation. To balance exploration, particularly if online queries differ from pretraining data, we set λ_a as the inverse of arm a's accuracy during the pretraining phase.

With this formulation, the posterior distribution of arm (LLM) embeddings at time t becomes: $p(\hat{\theta}_a^t|\mathcal{D}_t) = \mathcal{N}(\tilde{\theta}_a^t, (A_a^t)^{-1})$, where $\mathcal{D}_t$ represents the observed online data up to time t. The preference-prior informed LinUCB thus allows us to start with an informed estimate of LLM performance based on offline preference data, while still adapting to task dependent bandit feedback and query characteristics through online learning.

The algorithm balances exploration and exploitation by selecting the arm that maximizes the upper confidence bound, i.e. a_t is defined as:

$$\arg\max_a \Big(\cos(\hat{\psi}(q_t), \tilde{\theta}_a^t) + \alpha\sqrt{\hat{\psi}(q_t)^\top (A_a^t)^{-1} \hat{\psi}(q_t)} \Big)$$

where α is the exploration parameter.

To validate PILOT, we theoretically show that a preference-prior informed bandit algorithm can achieve a smaller regret bound than the standard algorithm. Here, we focus on Optimism in the face of uncertainty linear bandit algorithm (OFUL) (Abbasi-Yadkori et al., 2011a) since both OFUL & LinUCB are based on the same principle, i.e., principle of optimism in the face of uncertainty (Lattimore and Szepesvári, 2020, Chp 7.1) & OFUL is theoretically well-studied.

Proposition 2.1 (Validity of a preference-prior informed bandit (informal))**.** *Let $\theta^{\text{pref}} \in \mathbb{R}^{d'}$ and $\theta^* \in \mathbb{R}^{d'}$ be a pretrained vector and an unknown reward vector respectively. Let PI-OFUL be the OFUL (Abbasi-Yadkori et al., 2011a) with the Preference-prior-Informed initialization with θ^{pref}. Then, if $\|\theta^{\text{pref}} - \theta^*\| \leq \|\theta^*\|$, PI-OFUL achieves a smaller cumulative regret bound than OFUL.*

See Sec C.1 for a more formal statement & proof.

2.2.3 Enforcing Budget Constraint with Online Cost Policy

To manage user-specified cost budgets (B over Q queries), we introduce an online cost policy. This policy aims to optimally allocate the budget across

unseen queries to maximize expected reward. We frame this as an online multi-choice knapsack problem (ON-MCKP) (Chakrabarty et al., 2008).

This ON-MCKP formulation allows leveraging the ZCL algorithm (Zhou et al., 2008) to enforce budget constraints while maximizing expected reward (Equation 1). In ON-MCKP, a knapsack of capacity B receives item sets N_t over time; at most one item (with value v_j and weight w_j) is selected from each N_t to maximize total value within B. *In our context, at timestep t, available LLMs L form the item set; their reward estimates* $(\cos(\hat{\psi}(q_t), \hat{\theta}_l^t)$ $\forall l \in L)$ *are values, and estimated token costs are weights*. We assume known upper/lower bounds (UB, LB) on the reward-to-cost ratio and query costs small relative to B, standard for online problems (Zhou et al., 2008). The policy maintains budget utilization $z_t \in [0, 1]$. Following (Zhou et al., 2008), eligible LLMs $E_t \subset L$ must satisfy $C_t^l \leq \frac{\cos(\hat{\psi}(q_t), \hat{\theta}_l^t)}{(\frac{UB \cdot e}{LB})^{z_t}(\frac{LB}{e})}$. We select the LLM with the highest expected reward from E_t and update z_t.

Since the ZCL policy assumes an infinite horizon, potentially leading to underutilized budget over Q queries, we implement a binning strategy. The Q queries are partitioned into N bins of size S, where $N = \lceil \frac{Q}{S} \rceil$ bin budget $= \frac{B}{N}$. The cost policy is applied per bin, with unused budget spilling over to the next, allowing flexible allocation within overall constraints. Algorithm 3 (Appendix) details this policy, and Algorithm 2 provides an overview. Zhou et al. (2008, Theorem 5.1) establishes a performance guarantee, showing our online policy's performance is provably close to an optimal offline policy with full query knowledge.

Algorithm 1 PILOT (**P**reference-prior **I**nformed **L**inUCB f**O**r Adaptive Rou**T**ing)

Input: Human preference data D_{pref}, LLMs L

Preference-Based Pretraining

1: Learn query projection ϕ by minimizing triplet loss using $(q_a, l_i, l_j, l_{\text{win}})$ tuples from D_{pref} and constructing negative and positive samples as mentioned in Section 2.2.
2: Fix ϕ and learn LLM embeddings $\theta_{\text{LLM}}^{pref}$ using binary cross-entropy loss.

Online Bandit Learning

3: Initialize bandit learning parameters $A_a = \lambda_a I$ and $b_a = \lambda_a \theta_a^{pref}$ for all $a \in L$.
4: **for** $t = 1, \ldots, T$ **do**
5: Define a_t as arm/LLM with largest UCB.
6: Observe feedback r_t w.r.t. response of selected LLM a_t & update parameters A_a & b_a.
7: **end for**

Algorithm 2 Online Cost Policy

Input: budget B, query set length Q, LLMs L

1: **for** $t = 1, \ldots, Q$ **do**
2: Estimate token cost $C_t^l \ \forall \ l \in L$ for q_t.
3: Compute the cost eligibility threshold for each LLM l: $th_t^l = \frac{\cos(\hat{\psi}(q_t), \hat{\theta}_l^t)}{\left(\frac{UB \cdot e}{LB}\right)^{z_t} \cdot \left(\frac{LB}{e}\right)}$, where $z_t \in [0, 1]$ is the current budget utilization.
4: **yield** $l^* = \arg\max_{(l \in L \ \& \ C_t^l \leq th_t^l)} \cos(\hat{\psi}(q_t), \hat{\theta}_l^t)$
5: **end for**

3 Experimental Setup

3.1 Evaluation Details

To the best of our knowledge, we are the first to study LLM routing in an online bandit learning setting. Given the absence of an established experimental framework, we design the evaluation process by taking inspiration from (Li et al., 2010). Our objective is to simulate an online learning setting using an existing LLM routing dataset (Routerbench (Hu et al., 2024)). We first split the routing dataset into `tuning` data (for hyperparameter selection) and `evaluation` data. The objective of `evaluation` data is to simulate online user query traffic. Similar to news recommendation (Li et al., 2010), when deploying the bandit routers to users, one reasonable way is to split all traffic into two buckets - (i) "learning bucket": a fraction of traffic on which various bandit algorithms are run to learn. (ii) The other, called "deployment bucket", is where we greedily serve users using bandit router obtained from learning bucket.

3.1.1 Dataset

We evaluate our proposed method using Routerbench (Hu et al., 2024), a comprehensive LLM routing dataset spanning a wide range of tasks including commonsense reasoning, knowledge-based language understanding, conversation, math, and coding. Routerbench is constructed by leveraging existing datasets (MMLU, Hellaswag, GSM8k, Winogrande, ARC Challenge, MTBench, MBPP) commonly used to evaluate leading LLMs. The dataset comprises $36,497$ samples covering 64 tasks, with responses from 11 different LLMs, including both open-source (Llama-70B-chat, Mixtral-8x7B-chat, Yi-34Bchat , Code Llama-34B, Mistral-7B-chat,

WizardLM13B) and proprietary models (GPT-4, GPT-3.5-turbo, Claude-instant-v1, Claude-v1, Claude-v2). The dataset also includes the incurred cost and evaluation score (GPT-4 evaluation or exact match score, based on the task) of each LLM on each query. We request reader to see the original paper for details (Hu et al., 2024). We use ChatArena (Chiang et al., 2024) for preference data, sampling a subset of queries where both associated LLMs belonged to RouterBench's set of 11 LLMs.

3.1.2 Dataset Partitions

We partition the dataset as follows: we use 1000 samples as `tuning` data for hyperparameter selection, with the remaining data split into "learning" and "deployment" buckets with a 10:1 ratio. This setup allows us to observe the router's performance improvement as more data becomes available in the learning bucket over time.

3.1.3 Baselines

We want to primarily understand two questions, 1) how well our approach performs against existing routers, and 2) in context of using a bandit setting, how well the use of PILOT justifies against other choices of bandit algorithms.

For the first, we compare with *all-to-one* routers (all queries to a single LLM). Appendix D.1 also includes a reference comparison with *HybridLLM* (Ding et al., 2024), a supervised binary router, noting its different supervision requirement (optimal LLM index vs. our evaluative feedback). For the second, we use several contextual bandit baselines: *LinUCB* (Li et al., 2010) (UCB with a linear reward model), *Epoch-Greedy* (Langford and Zhang, 2007) (alternating exploration/exploitation), *Explore Only* (continuous exploration), and a *Random Policy* (random LLM selection). Crucially, for fair comparison of allocated budget against deployment performance (Figures 3, 5), our proposed cost policy is uniformly applied across all baselines. The policy's effectiveness is further assessed in Fig 4.

3.2 Implementation Details

Embedding Model. We use OpenAI's `text-embedding-3-small` to embed queries for all results shown in Figure 3. To analyze the sensitivity of our router, we conduct an experiment using `Instructor-XL` (Su et al.) in Section 5.4.

Hyperparameter selection. We use the tuning data to fine-tune hyperparameters of our method and baselines. To optimize the exploration parameter α (for PILOT and LinUCB), we search over $\{1, 1.5, 2, 5, 10\}$, selecting the value that maximizes reward. For Epoch Greedy, we do a grid search over window sizes $\{10, 50, 100, 500\}$ to find the optimal window size.

Online Cost Policy. During the deployment phase, our cost policy requires estimating the total cost of each query, including both input and output token costs. Input tokens are determined from the query itself, while output tokens are estimated using the mean output token count from responses in the tuning data for each LLM. This mean is then applied to all queries in the deployment set to calculate total query costs. Furthermore, It is worth noting that the proposed online cost policy operates independently of the PILOT algorithm. Its objective is to select the most suitable LLM for each query based on a query-wise LLM ranking, aiming to maximize the cumulative reward across Q queries while adhering to the total budget B.

4 Results and Analysis

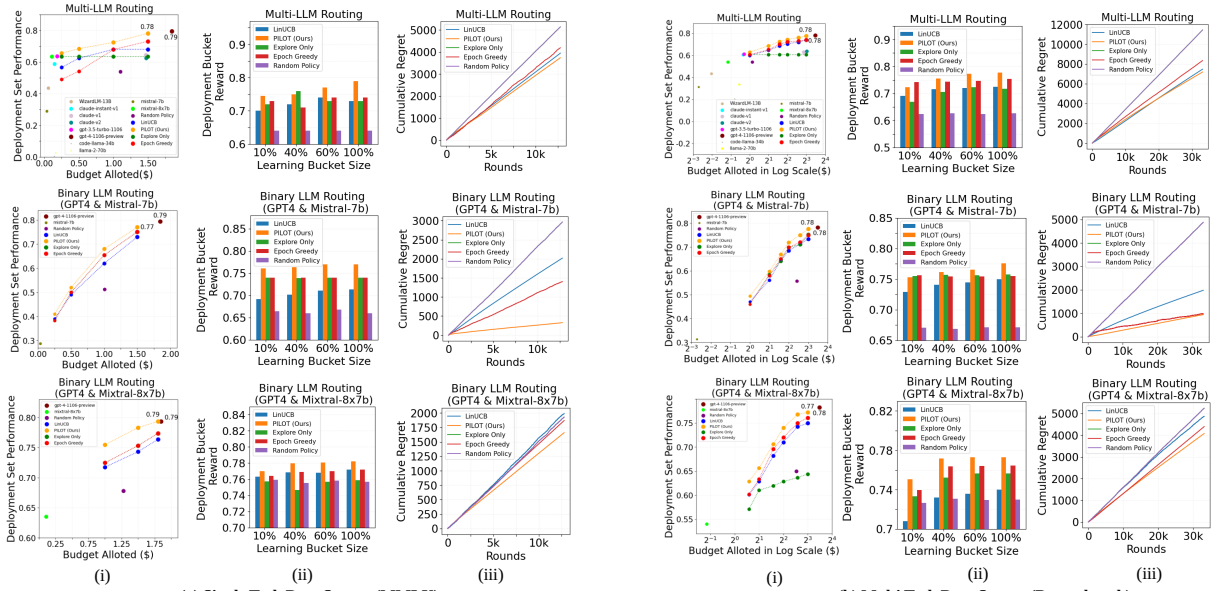
We evaluate PILOT's performance across various facets of LLM routing, considering diverse, multi-task applications and specific use cases. Our experiments utilize two data types:

(i) *Single-task data source*: The MMLU benchmark from Routerbench, focusing on multi-choice question answering. (ii) *Multi-task data source*: Full Routerbench dataset, encompassing tasks like code generation, math problems, & multi-turn conversations, simulating a broad range of user queries.

Using these datasets, we investigate several aspects of our bandit-based LLM routing algorithm, detailed under *Main Experiments* in Table 3. Below, we summarize PILOT's performance for both single and multi-task scenarios, covering multi-LLM and binary-LLM routing settings. For binary-LLM routing, Figure 3 presents results for two LLM pairs, with more comparisons in Appendix D.4.

Multi-Task Data Source Setting Results in Figure 3 column *b*, indicate our method's superiority: We achieve a performance equal to 93% of GPT-4's at just 25% of its cost in multi-LLM setting while maintaining higher performance than any other baselines. Also, our method consistently shows the highest deployment set performance & lower regret across various learning conditions.

Single-Task Data Source Setting As shown in column *a* of Figure 3, our method (PILOT) consistently outperforms all baseline bandit algorithms. In particular, in the multi-LLM routing case, we at-



(a) Single Task Data Source (MMLU) (b) Multi Task Data Source (Routerbench)

Figure 3: **Bandit Feedback based LLM Routing Evaluation**: In column (a) we report results for single task data source setting (MMLU), and in column (b) we report results for multi-task data source setting (Routerbench). The sub-column (i) in each column represents performance vs cost curves on the held-out deployment set; sub-column (ii) represents performance across different learning bucket sizes; sub-column (iii) represents cumulative regret. Our method is shown in orange.

tain a performance of 86% of GPT-4 at only 27% of its cost and surpass all other all-to-one LLM baselines. PILOT also exhibits highest performance on deployment set across various learning bucket sizes, showing its efficacy with limited data.

5 Discussion and Ablations

This section provides qualitative and quantitative analyses of PILOT's routing behavior, computational efficiency, cost policy, and sensitivity, offering a holistic view. These analyses, summarized under '*Analysis*' in Table 3, delve deeper into PILOT's operational characteristics.

5.1 Qualitative Analysis of PILOT's Routing

Qualitative examination of PILOT's routing reveals intelligent decision-making. For demanding tasks like MMLU and ARC Challenge, PILOT routes 90% and 89.4% of queries to GPT-4, respectively, leveraging GPT-4's strength in complex reasoning. For coding tasks (MBPP), while GPT-4 is utilized, Claude models handle a significant 28% of queries, indicating PILOT's recognition of Claude's coding abilities. In GSM8K, Claude (v1) is the predominant choice (94% of queries). This isn't arbitrary; Claude-v1's strong performance on math tasks, coupled with its lower cost, makes it an effective option, echoing findings in Zhang et al.

5.2 Analysis of the Online Cost Policy

Comparison with other Online Cost Policies: Here we compare it to simple online baselines: (i) allocating $\frac{B}{Q}$ budget per query, and (ii) allocating $\frac{B}{Q}$ per query with spillovers from previous queries. We use these query-wise budgets to select the highest-ranked arm/LLM within the budget. We evaluate using mean reciprocal rank of the chosen arm (where rank 1 is the best arm w.r.t. learned bandit router) and performance on Routerbench's deployment set (multi-LLM case).

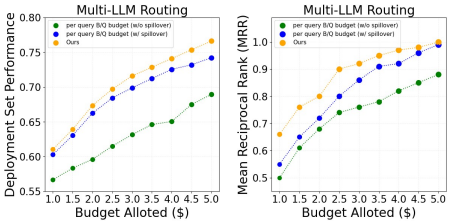


Figure 4: **Cost Policy Comparison**: (Left) Mean Reciprocal Rank of the chosen arms for various budgets (Right) Performance of cost policies with diff budget

Comparison with Offline Cost Policy: Here, we try to maximize $P - \lambda C$ (following Chen et al., 2022), where P is estimated performance, C is cost, & hyperparameter λ is tuned retrospectively for each budget constraint by optimizing its value

Cost ($)	$P - \lambda\mathcal{C}$	PILOT (Ours)	Difference
0.25	0.6079	0.6557	**+0.0478**
0.50	0.6602	0.6840	**+0.0238**
1.00	0.7265	0.7240	-0.0025
1.50	0.7740	0.7814	**+0.0073**

Table 1: Comparison of performance between $P - \lambda\mathcal{C}$ offline policy and PILOT's online cost policy

over entire deployment set to achieve best possible outcome for that budget-a significant informational advantage. As shown in Table 1, our adaptive online policy generally outperforms this offline policy across multiple cost thresholds. This shows value of our online approach which performs comparably / better than even a policy with perfect hindsight.

5.3 Computational Overhead of Routing

Embedding Model	Routing Time	GPT-4 Inference Time
`Instructor-XL`	0.065 s	2.5 s
OpenAI `text-embedding-3-small`	0.239 s	2.5 s

Table 2: **Analysis of Routing Time**: Routing Time refers to average time taken by PILOT to select a LLM from the pool and GPT-4 Inference Time is average time taken by GPT-4 to answer a query on MMLU dataset

To assess PILOT's efficiency, we compare its average LLM selection time against GPT-4's average inference time on MMLU (Table 2). PILOT's routing time is **10x** and **38x** faster than GPT-4 inference when using Instructor-XL & OpenAI embeddings, respectively. This shows that PILOTadds negligible overhead to the response generation pipeline.

5.4 Embedding Model Sensitivity

Here, we assess PILOT's sensitivity to the embedder by evaluating its performance using another model - `Instructor-XL` (Su et al., 2023). As shown in Figure 5, PILOT continues to maintain superior performance over baselines.

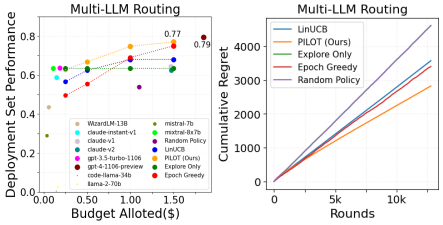


Figure 5: **Embedding Sensitivity Analysis** Figure compares PILOT's performance with bandit baselines using `Instructor-XL` embedding.

6 Related Works

Research on efficient LLM deployment spans static model optimizations, hybrid strategies, and dynamic routing. (1) *Static efficiency methods*, such as pruning (LeCun et al., 1989), quantization (Jacob et al., 2018), LoRA (Hu et al., 2021), and distillation (Hinton et al., 2015), compress models for fixed cost constraints, but cannot adapt to varying task demands. (2) *Hybrid approaches* like LLM-Blender (Lu et al., 2024) synthesize outputs from multiple LLMs to improve quality, while others such as TensorOpera Router (Stripelis et al., 2024b) and FORC (Šakota et al., 2024) train offline meta-models to predict the best model. These require full supervision and may not generalize to new query distributions. (3) *Routing strategies* select a single model per query. Non-predictive methods like FrugalGPT (Chen et al., 2023) use sequential evaluation, while predictive ones like HybridLLM (Ding et al., 2024), GraphRouter (Feng et al., 2024), and Confidence Tokens (Chuang et al., 2025) train supervised routers with full feedback or LLM modifications. While MetaLLM (Nguyen et al., 2024) models LLM routing as a contextual bandit problem with a fixed cost-performance trade-off, it doesn't offer strict budget guarantees. OptLLM (Liu et al., 2024b) treats routing as a multi objective optimization problem with the goal to optimize cost and performance, but it relies on full supervision i.e. a training set generated by answering every query using all LLMs, which is infeasible in practical settings due to the cost of generation. The same drawback is also applicable to TensorOpera Router (Stripelis et al., 2024a) which proposes a multi-phase pipeline for data generation, training and deployment of routers.

In contrast, our work formulates routing as a contextual bandit problem with budget constraints. We learn an embedding-based router that adapts online using bandit feedback—observing reward only for the selected model—without requiring exhaustive supervision or query-specific full inference. This enables efficient and adaptive LLM deployment in dynamic, cost-sensitive environments.

7 Conclusion

We address LLM routing with budget constraints using bandit feedback in this work. We propose PILOT, a human-preference prior based contextual bandit algorithm, coupled with a novel online cost policy that optimizes budget allocation across

queries. Our approach achieves 93% of GPT-4's performance at 25% of its cost on Routerbench.

Limitations

During the online bandit learning (Algorithm 1) we do not consider budget constraint, rather only during deployment we consider budget constraint. Underlying rational for decoupling the bandit algorithm and the cost policy was to ensure deployment stability and provide direct, user-controllable budget management. This separation facilitates deploying a robust bandit model while dynamically adjusting cost policy in real-time based on budget. However, one maybe be interested in learning under budget constraints, which we leave for future work.

In this work we focused on single-turn conversations as input for routing, however real-world scenarios could multi-turn interaction based routing. We leave this for future work.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011a. Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems*, 24.

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011b. Online least squares estimation with self-normalized processes: An application to bandit problems. *arXiv preprint arXiv:1102.2670*.

Appcues. 2024. Rating systems in ux: Star vs. thumbs up. *Appcues Blog*.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. 2024. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*.

Djallel Bouneffouf and Irina Rish. 2019. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Deeparnab Chakrabarty, Yunhong Zhou, and Rajan Lukose. 2008. Online knapsack problems. In *Workshop on internet and network economics (WINE)*, pages 1–9.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and 1 others. 2024. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*.

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

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*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *Preprint*, arXiv:2403.04132.

Yu-Neng Chuang, Helen Zhou, Prathusha Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, and Xia Hu. 2025. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 3.

Delighted. 2024. 7 types of customer experience surveys you should know about. *Delighted Blog*.

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*.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2024. Graphrouter: A graph-based router for llm selections. *arXiv preprint arXiv:2410.03834*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv e-prints*, pages arXiv–2106.

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*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. 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)*, pages 14165–14178.

John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in neural information processing systems*, 20(1):96–1.

Tor Lattimore and Csaba Szepesvári. 2020. *Bandit algorithms*. Cambridge University Press.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. 2024. Llm inference serving: Survey of recent advances and opportunities. *arXiv preprint arXiv:2407.12391*.

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.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024a. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

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

Xiaoding Lu, Adian Liusie, Vyas Raina, Yuwen Zhang, and William Beauchamp. 2024. Blending is all you need: Cheaper, better alternative to trillion-parameters llm. *arXiv preprint arXiv:2401.02994*.

Quang H Nguyen, Thinh Dao, 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. *Preprint*, arXiv:2406.18665.

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.

Dimitris Stripelis, Zijian Hu, Jipeng Zhang, Zhaozhuo Xu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Salman Avestimehr, and Chaoyang He. 2024a. Tensoropera router: A multi-model router for efficient llm inference. *Preprint*, arXiv:2408.12320.

Dimitris Stripelis, Zhaozhuo Xu, Zijian Hu, Alay Dilipbhai Shah, Han Jin, Yuhang Yao, Jipeng Zhang, Tong Zhang, Salman Avestimehr, and Chaoyang He. 2024b. Tensoropera router: A multi-model router for efficient llm inference. In *EMNLP (Industry Track)*.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121.

Hugh Zhang, Jeff Da, Dean Lee, Vaughn Robinson, Catherine Wu, William Song, Tiffany Zhao, Pranav Raja, Charlotte Zhuang, Dylan Slack, and 1 others. 2024. A careful examination of large language model performance on grade school arithmetic. *Advances in Neural Information Processing Systems*, 37:46819–46836.

Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. 2008. Budget constrained bidding in keyword auctions and online knapsack problems. In *Proceedings of the 17th international conference on world wide web*, pages 1243–1244.

A Appendix

In this section, we provide additional results and details that we could not include in the main paper due to space constraints. In particular, this appendix contains the following:

B Extended Related Works

In Section 6 of the main paper we briefly described works in the areas of routing strategies for language models and neighbouring research topics such as efficient LLM inference and hybrid LLM approaches. Here we elaborate on it.

B.1 Efficient LLM Inference

Traditional approaches to efficient ML inference can be categorized into: model pruning (LeCun et al., 1989), quantization (Jacob et al., 2018), linear attention (Arora et al., 2024), low-rank adaptation (Hu et al., 2021) and knowledge distillation (Hinton et al., 2015). These methods typically produce static optimizations, which may not suffice for LLMs serving a range of tasks with varying accuracy/cost constraints. Our work, in contrast, focuses on dynamic optimizations to meet diverse user demands.

B.2 Hybrid LLM Approaches

Hybrid inference methods attempt to balance cost and quality by combining outputs or using meta-models. These include:
(i) *Multi-LLM Synthesis*: LLM-Blender (Lu et al., 2024) and related methods (Jiang et al., 2023) invoke several models and fuse their responses. While improving output quality, these approaches are cost-intensive and unsuitable for latency-sensitive applications.
(ii) *Meta-routing via reward estimation*: TensorOpera Router (Stripelis et al., 2024b) builds a separate reward model to guide routing decisions over multiple LLMs. However, it relies on offline data and full supervision to train the reward predictor.
Unlike these works, we focus on selecting a single model per query and improve performance through online learning from partial feedback.

B.3 Routing Strategies for LLMs

Existing routing strategies can be categorized as:
(i) *Non-predictive routing*: FrugalGPT (Chen et al., 2023) executes models sequentially until a quality threshold is met. While simple, this leads to multiple model calls and doesn't generalize across queries.
(ii) *Predictive routing (supervised)*: These methods train a router to choose among LLMs based on full supervision. HybridLLM (Ding et al., 2024) trains a classifier to select between a small and large model. FORC (Šakota et al., 2024) uses a meta-model to balance accuracy and cost. GraphRouter (Feng et al., 2024) encodes queries and models into a bipartite graph to guide routing. Chuang et al. (Chuang et al., 2025) introduce confidence tokens emitted by LLMs to help with model selection, requiring LLM modification.
While these methods show strong performance, they rely on full model evaluation during training, limiting scalability and adaptability. Our work differs by treating routing as a contextual bandit problem: we learn only from the selected model's feedback, adapting online to shifting distributions without exhaustive supervision.
Below we summarize our unique aspects to contextualize our work within LLM routing literature:

- **Online Adaptation**: We train the router online, enabling it to adjust to evolving query types and workloads.
- **Bandit Feedback**: We operate under partial supervision, learning from the reward of only the selected model, unlike prior work requiring all-model inference for each training query.
- **Budget-Aware Routing**: Our formulation includes a cost policy via a multi-choice knapsack, explicitly managing user budgets.

This makes our method suitable for practical LLM deployment settings that demand efficiency, adaptability, and minimal supervision.

Experiment	Details	Reference
	MAIN EXPERIMENTS	
Performance vs Budget Curves	Evaluates performance-cost trade-offs across budget constraints.	*Fig. 3 (a), (b)(i)*
Performance with Varying Learning Data Sizes	Measures performance with different learning data quantities in deployment.	*Fig. 3 (a), (b)(ii)*
Cumulative Regret Curves	Tracks learning efficiency over time compared to baselines.	*Fig. 3 (a), (b)(ii)*
	ANALYSIS	
Qualitative Analysis of Routing	Examines routing decisions across diverse tasks (MMLU, MBPP, GSM8K).	*Section 5.1*
Compute Overhead of Routing	Measures routing latency overhead against GPT-4 inference time.	*Section 5.3*
Online Cost Policy Analysis	Compares adaptive online policy with fixed-budget and offline policies.	*Section 5.2*
Embedding Model Sensitivity	Tests robustness across different query embedding models.	*Section 5.4*
Comparison with Static Binary Supervised LLM Router	Contrasts with static supervised routers, especially under distribution shifts.	*Appendix D.1*
Human Preference Learning	Evaluates the human preference learning stage detailed in Section 2.2.1.	*Appendix D.3*
Ablation & Sensitivity Analysis	Analyzes impact of individual components and robustness factors.	*Appendix D.6*

Table 3: **Overview of Experiments:** Summary of empirical evaluations conducted to assess PILOT's performance.

C Theoretical Analysis and Algorithms

C.1 Theoretical Analysis of a Preference-Prior-Informed Bandit Algorithm

To show the validity of PILOT, we theoretically show that a preference-prior-informed bandit algorithm can achieve a smaller regret bound than the standard algorithm. Here, we focus on preference-prior informed OFUL (Abbasi-Yadkori et al., 2011b) since OFUL is a theoretically well-studied method, and both the algorithms (LinUCB and OFUL) share the same principle, i.e., the principle of optimism in the face of uncertainty (Lattimore and Szepesvári, 2020, Chp 7.1) and (Li et al., 2010) have not provided a theoretical analysis of LinUCB. We briefly introduce a problem setting for theoretical analysis. For $t = 1, \dots, T$, a query q_t and selected LLM l_t, we assume the following reward model:

$$r_t(l_t, q_t) = \theta^* \cdot x(l_t, q_t) + \varepsilon_t,$$

where $x(l_t, q_t) \in \mathbb{R}^{d'}$ is a context vector and $\theta^* \in \mathbb{R}^{d'}$ is an unknown reward vector. We define cumulative regret

$$R(T) = \sum_{t=1}^{T} \left(\max_{l \in L} r_t(l, q_t) - r_t(l_t, q_t) \right).$$

For $\lambda > 0$, we define an estimation $\hat{\theta}_t$ of θ^* as $A_t^{-1} \sum_{s=1}^{t-1} r_s x(l_s, q_s)$, where $A_t = \lambda I + \sum_{s=1}^{t-1} x(l_s, q_s) x(l_s, q_s)^\top$. For $\delta \in (0, 1)$ and $S > 0$, we define a confidence set $\mathcal{C}_t(\delta, \hat{\theta}_t; S)$ by

$$\mathcal{C}_t(\delta, \hat{\theta}_t; S) = \Big\{ \theta \in \mathbb{R}^{d'} : (\theta - \hat{\theta}_t)^\top A_t (\theta - \hat{\theta}_t) \leq \sqrt{\lambda} S + R \sqrt{2 \log(1/\delta) + d' \log \left(1 + \frac{T}{\lambda d'} \right)} \Big\}.$$

Then for each round t, OFUL selects $l_t \in L$ such that $\max_{\theta \in \mathcal{C}_t(\delta, \hat{\theta}_t; S)} \theta \cdot x(l_t, q_t) = \max_{(\theta, l) \in \mathcal{C}_t(\delta, \hat{\theta}_t; S) \times L} \theta \cdot x(l, q_t)$.

Then, if $\|\theta^*\| \leq S$, OFUL has the following regret bound with probability at least $1 - \delta$,

$$R(T) \leq U_T(S)$$

where

$$U_T(S) = 4 \sqrt{T d' \log(\lambda + T/d')} \cdot \left(\sqrt{\lambda} S + R \sqrt{2 \log(1/\delta) + d' \log \left(1 + \frac{T}{\lambda d'} \right)} \right)$$

If we know the optimal choice of S (i.e., $S = \|\theta^*\|$), then regret bound $U_T(\mathrm{OFUL})$ of OFUL is given as

$$U_T(\mathrm{OFUL}) := U_T(\|\theta^*\|).$$

Proposition C.1. *Let $\theta^{\mathrm{pref}} \in \mathbb{R}^{d'}$ be a pre-trained vector. We define an estimation $\tilde{\theta}_t$ of θ^* with an initialization θ^{pref} as $\tilde{\theta}_t := (A_t)^{-1} b_t$, where $A_t = \lambda I + \sum_{s=1}^{t-1} x(l_s, q_s) x(l_s, q_s)^\top$, $b_t = \lambda \theta^{\mathrm{pref}} + \sum_{s=1}^{t-1} x(l_s, q_s)$. We define PI-OFUL as OFUL with the confidence set $\mathcal{C}_t(\delta, \tilde{\theta}_t; S')$ with a parameter $S' > 0$, that is, PI-OFUL selects $l_t \in L$ such that $\max_{\theta \in \mathcal{C}_t(\delta, \tilde{\theta}_t; S')} \theta \cdot x(l_t, q_t) =$*

$\max_{(\theta,l)\in\mathcal{C}_t(\delta,\tilde{\theta}_t;S'))\times L} \theta\cdot x(l,q_t)$. *If* $\|\theta^*-\theta^{\text{pref}}\|\leq S'$*, then regret bound of PI-OFUL is given as* $U_T(S')$*. In particular, if we know the optimal choice of* S' *(i.e.,* $S'=\|\theta^*-\theta^{\text{pref}}\|$*), then then regret bound* $U_T(\text{PI-OFUL})$ *of PI-OFUL is given as*

$$U_T(\text{PI-OFUL}) := U_T(\|\theta^*-\theta^{\text{pref}}\|).$$

Thus, if $\|\theta^*-\theta^{\text{pref}}\|\leq\|\theta^*\|$*, we have*

$$U_T(\text{PI-OFUL})\leq U_T(\text{OFUL}).$$

Proof. For $1\leq t\leq T$, let $X\in\mathbb{R}^{(t-1)\times d'}$ be the matrix whose rows are given as $x(a_1,q_1)^\top,\ldots,x(a_{t-1},q_{t-1})^\top$. By definition of $\tilde{\theta}_t$ and the proof of (Abbasi-Yadkori et al., 2011b, Theorem 8) and, $\tilde{\theta}_t$ is given as

$$\tilde{\theta}_t=(X^\top X+\lambda I)^{-1}X^\top\varepsilon+\theta^*-\lambda(X^\top X+\lambda I)(\theta^*-\theta^{\text{pref}}),$$

where $\varepsilon\in\mathbb{R}^{t-1}$ is defined as $\varepsilon^\top=(\varepsilon_1,\ldots,\varepsilon_{t-1})$. Thus, by the proof of (Abbasi-Yadkori et al., 2011b, Theorem 8), we have the following with probability $1-\delta$

$$|\tilde{\theta}_t\cdot x-\theta^*\cdot x|\leq \|x\|_{A_t^{-1}}\left(R\sqrt{2\log\left(\frac{\det(A_t)^{1/2}\det(\lambda I)^{-1/2}}{\delta}\right)}+\sqrt{\lambda}\left\|\theta^*-\theta^{\text{pref}}\right\|\right).$$

Using this confidence bound, by the standard argument (the same proof as (Abbasi-Yadkori et al., 2011b, Theorem 13)), we have our assertion. □

C.2 Algorithm for the Online Cost Policy

In Section 2.2.3 of the main paper, we introduced our online cost policy, enabling users to set a cost budget B and distribute it across a defined number of queries Q. Building on that discussion, we present the corresponding algorithm in Algorithm 2 and provide a concise summary of its key steps below. The algorithm operates by dividing the total query budget Q into N bins of size S, where $N=\lceil\frac{Q}{S}\rceil$. Each bin is allocated a portion of the total budget B, denoted as $B_{\text{bin}}=\frac{B}{N}$. At the start of each bin, the algorithm adds B_{bin} to the remaining budget B_{left} (budget leftover from previous bins/timesteps). For each query within a bin, the algorithm selects eligible large language models (LLMs), E, from the set L whose costs C^l $(l\in L)$ are below a threshold. This threshold is sensitive to both the LLM l and the current budget utilization z, and is given by $\frac{\cos(\hat{\psi}(q_t),\hat{\theta}_l^t)}{(\frac{UB.e}{LB})^z(\frac{LB}{e})}$. If no LLMs fit the allotted budget thresholds, the algorithm adjusts the budget per query to $\frac{B_{\text{left}}}{Q_{\text{left}}}$ (Q_{left} is the number of queries remaining in the bin) and re-evaluates eligible LLMs. If still no LLMs are available, the algorithm terminates with an "Insufficient budget" message. Otherwise, it selects the LLM with the highest expected reward (defined in Equation 1) in the set of eligible LLMs E, updates the remaining budget, and yields the selected LLM for the current query. This process repeats for each query in the bin, ensuring the budget is optimally utilized across the entire query set.

Algorithm 3 Online Cost Policy

Require: Budget B, # queries Q, Bin size S, LLM set L
Ensure: LLM selections for each query
  $N\leftarrow\lceil\frac{Q}{S}\rceil$ % Number of bins
  $B_{\text{bin}}\leftarrow\frac{B}{N}$ % Budget per bin
  $B_{\text{left}}\leftarrow 0$
  for each bin i in 1 to N **do**
    $z\leftarrow 0$ % Initialize budget utilization
    $B_{\text{left}}\leftarrow B_{\text{left}}+B_{\text{bin}}$ % Budget Left
    for each query q_t in bin i **do**
      $Q_{\text{left}}\leftarrow$ number of remaining queries in bin
      $E\leftarrow\{l\in L: C^l\leq\frac{\cos(\hat{\psi}(q_t),\hat{\theta}_l^t)}{(\frac{UB.e}{LB})^z(\frac{LB}{e})}\}$
      if E is empty **then**
        $B'\leftarrow\frac{B_{\text{left}}}{Q_{\text{left}}}$ % Adjusted budget
        $E\leftarrow\{l\in L: C^l\leq B'\}$
        if E is empty **then**
          return "Insufficient budget"
        end if
      end if
      $l^*\leftarrow\arg\max_{l\in E}\cos(\hat{\psi}(q_t),\hat{\theta}_l^t)$ % pick best LLM
      $B_{left}\leftarrow B_{left}-C^l$ % Update budget
      $z\leftarrow z+\frac{C^l}{B_{\text{bin}}}$ % Update budget use
      yield l^* % Return selected LLM
    end for
  end for

D Additional Experimental Results and Analysis

D.1 Comparison with Supervised Binary LLM Router

As stated in Section 3.1.3, here (in Figure 6) we compare our bandit router PILOT with the state-of-

the-art supervised router HybridLLM, while keeping in mind that HybridLLM (Ding et al., 2024) relies on full supervision, whereas our approach operates with only bandit feedback arriving online. Furthermore, we quantitatively evaluate PILOT and HybridLLM, w.r.t. adaptability to shift in user queries to understand whether PILOT we in

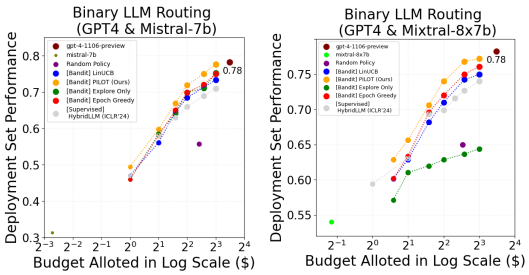


Figure 6: **Performance vs Cost comparison with Supervised HybridLLM (Ding et al. 2024)** The left figure shows binary LLM routing comparisons for GPT-4 and Mistral-7b in the LLM pool. The right figure presents similar comparisons, this time for GPT-4 and Mixtral-8x7b. This study uses the Routerbench dataset

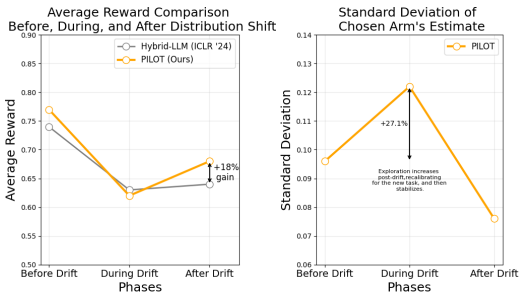


Figure 7: **Adaptability to Shift in Query Distribution** The left figure shows average reward comparison across different time instants - "Before", "During" and "After" shift in query distribution from MMLU to GSM8k. The right figure shows that during the drift the exploration increases as new distribution is encountered and then the exploration settles down.

D.1.1 Performance vs Cost Analysis

We compare PILOT with the deterministic variant of HybridLLM (Ding et al., 2024), which assumes that LLMs are deterministic functions mapping input features to a single point in the output space. We do not experiment with the probabilistic variants due to the high cost involved—they require 20 times more LLM calls to train the router, making them expensive to implement. These probabilistic variants utilize soft labels for training rather than hard labels, which necessitates sampling 10 responses from each LLM (thus 20 calls for two LLMs) per query and calculating a sample average. We use the Routerbench dataset for this comparison and pick two LLM combinations (GPT4 & Mistral-7b, and, GPT4 & Mixtral-7x8b) for binary routing task. We pick these LLM combinations to compare routers in the presence of large (GPT-4), medium (Mixtral-8x7b) and small (Mistral-7b) size language models. Furthermore, as per the protocol in Section 3.1 of the main paper, we use 1000 samples for hyperparamter tuning and the remaining samples is split into "learning" and "deployment" buckets with $10:1$ ratio.

As can be seen from Figure 6, PILOT performs on par with, and occasionally surpasses, HybridLLM. This underscores the effectiveness of bandit routers, achieving strong results without requiring full supervision.

D.1.2 Adaptability to Shift in Query Distribution

Here in Figure 7, we simulate a task shift (from MMLU to GSM8k) to create a streaming dataset, tracking average reward & exploration (standard deviation of chosen arm's estimate). We then compare rewards before ("Before"), at ("During") and 5000 steps after the transition ("After"). As can be seen in Figure 7 (left), PILOT adapts significantly post-drift, unlike the static supervised baseline. We also observe in Figure 7 (right) that the exploration of PILOT increases during drift and subdues after it, as expected.

D.2 Query Complexity Analysis

In the context of binary LLM routing, an effective routing algorithm should allocate more complex queries to the more capable model (GPT-4) while routing simpler queries to the less expensive model (Mistral-7B/Mixtral-8x7B) (Ding et al., 2024), to optimize performance within the given budget constraints. We thus investigate how our algorithm routes queries of varying complexity when faced with such a binary choice. For this analysis, we fix the overall budget for routing to $\$4$ and examine the average complexity of queries directed to each LLM. To quantify query complexity, we use *Evol Complexity* (Liu et al., 2024a) measure, which is useful for selecting hard samples for LLM alignment in comparison to scores such as LLM response *perplexity* and *Direct Scoring* (Chen et al., 2024).

The average query complexity of GPT-4 routed queries (QC) in Table 4 indicates that on average PILOT routes more complex queries to GPT-4, than

LLM Pool	LinUCB		PILOT	
	QC	p-value	QC	p-value
GPT4 - Mistral7B	2.61	0.06	**2.64**	**0.004**
GPT4 - Mixtral8x7B	2.62	0.05	**2.75**	**5e-37**

Table 4: **Query Complexity Analysis for Routed Queries:** *QC* refers to average complexity of GPT-4 routed queries, &, *p-value* is from a Mann-Whitney U Test on average query complexity scores obtained from LLMs in the pool.

LinUCB. Furthermore, the difference between the average query complexity of GPT-4 routed queries and Mistral-7B/Mixtral-8x7B routed queries is statistically significant for PILOT (Mann-Whitney U test's p-value < 0.05), unlike LinUCB. This indicates that PILOT not only considers the budget constraints but also assesses query complexity to make informed routing decisions.

D.3 Analysis of Human Preference Learning

Next, we analyze our offline human preference learning algorithm's (Section 2.2.1) accuracy of predicting the human preferred LLM for a given query. For this analysis, we use a subset of 500 samples from the ChatArena (Chiang et al., 2024) dataset (related to the 11 LLMs in Routerbench) that was not seen during training. We compare our accuracy with RouteLLM (Ong et al., 2024) which also uses human preference data for learning how to route. We find our algorithm achieves a higher accuracy of **65.0**, in comparison to RouteLLM's (uses Matrix factorization) accuracy of 63.6.

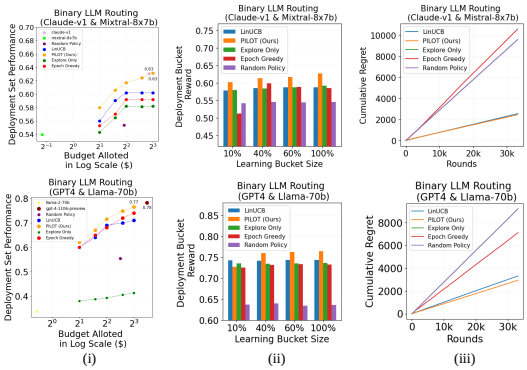
D.4 Binary LLM Routing Results

In Figure 3 of the main paper, we reported results for binary routing with two LLM combinations - GPT4 & Mistral-7b, and GPT4 & Mixtral-7x8b. Here, in Figure 8, we report results with two more LLM combinations (GPT4 & Llama2-70b, Claude v1 & Mixtral – 7x8B) to further evaluate our methods performance.

Similar to our observation in the main paper, we find that across metrics, PILOT performs better than baselines.

D.5 Sensitivity to Noise

We assess robustness of PILOT to noisy bandit feedback by conducting a synthetic noise experiment on MMLU, where the deployment set budget is fixed at \$1. Since the reward in MMLU is binary, we employ the following noise model: with some fixed probability $\epsilon \in [0, 1]$, the true binary



(i) (ii) (iii)

Figure 8: **Bandit Feedback based Binary LLM Routing Evaluation**: Figure reports results for multi task data source setting (Routerbench). In the top row we have Claude-v1 and Mixtral-7x8b LLMs in the LLM pool, and in the botton row we have GPT-4 and Llama2-70b LLMs in the LLM pool. Column (i) represents performance vs cost curves on the held-out deployment set; Column (ii) represents performance across different learning bucket sizes; Column (iii) represents cumulative regret. Our method PILOT , shown in orange, performs the best across the metrics.

reward (0 or 1) is replaced by a uniformly random binary value, that is, either 0 or 1 with equal probability. The results in Table 5 show that for PILOT, even at 5% noise, the reward drops by less than 4%, demonstrating strong resilience.

Noise Percentage (%)	Avg. Deployment Set Reward
0	0.761
1	0.750
3	0.740
5	0.730

Table 5: **Effect of Noise:** Average deployment set reward on MMLU datsaset across different noise levels.

D.6 Ablation and Sensitivity Analysis

In Figure 3 of the main paper, we analyzed the PILOT's performance of across cost budgets, and reward & regret across rounds of online learning. Here we study the goodness of our pre-trained router (preference data based router) on the Routerbench deployment set. Please note, the preference data, Chatarena (Chiang et al., 2024), is different from the Routerbench dataset (Hu et al., 2024). Next, we also study the effect of exploration parameter (used in UCB computation in PILOT and LinUCB) on reward on the tuning dataset.

D.6.1 Ablation Analysis

Here, in Table 6 the initial pre-trained router shows strong foundational performance, and incorporat-

Bud.	Pre-trained Router	Pre-trained + 10% Online	PILOT	LinUCB
$1	0.34	0.61	0.63	0.60
$1.5	0.34	0.61	0.66	0.63
$2	0.35	0.63	0.69	0.64
$3	0.38	0.65	0.73	0.68

Table 6: **Ablation Study:** Performance vs. Cost Comparison for pretrained router and effect of online training

α	PILOT	LinUCB
10	0.600	0.601
5	0.610	0.623
2	0.645	0.649
1	0.641	0.640

Table 7: **Sensitivity Analysis of** α

ing just 10% online data enables it to adapt quickly and approach the performance of LinUCB, especially under constrained budgets. Notably, PILOTconsistently outperforms the other approaches, even in low-cost scenarios.

D.6.2 Sensitivity Analysis

Table 7 presents the effect of the exploration parameter α on average reward, evaluated on the tuning dataset described in Section 3.1. Consistent with findings by (Li et al., 2010), the results exhibit an inverted U-shape. Small values of α result in insufficient exploration, limiting the algorithm's ability to discover optimal LLM-query matches. In contrast, excessively high α values lead to over-exploration, missing opportunities to maximize reward.