NEURAL BANDIT BASED OPTIMAL LLM SELECTION FOR A PIPELINE OF TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

With the increasing popularity of large language models (LLMs) for a variety of tasks, there has been a growing interest in strategies that can predict which out of a set of LLMs will yield a successful answer at low cost. This problem promises to become more and more relevant as providers like Microsoft allow users to easily create custom LLM "assistants" specialized to particular types of queries. However, some tasks (i.e., queries) may benefit from breaking down the task into smaller subtasks, each of which can then be executed by a LLM expected to perform well on that specific subtask. For example, in extracting a diagnosis from medical records, one can first select an LLM to summarize the record, select another to validate the summary, and then select another, possibly different, LLM to extract the diagnosis from the summarized record. Unlike existing LLM selection or routing algorithms, this setting requires selecting a sequence of LLMs, with the output of each LLM feeding into the next and potentially influencing its success. Thus, unlike single LLM selection, the quality of each subtask's output directly affects the inputs, and hence the cost and success rate, of downstream LLMs, creating complex performance dependencies that must be learned during selection. We propose a neural contextual bandit-based algorithm that trains neural networks that model LLM success on each subtask in an online manner, thus learning to guide the LLM selections for the different subtasks, even in the absence of historical LLM performance data. Experiments on telecommunications question answering and medical diagnosis prediction datasets illustrate the effectiveness of our proposed approach compared to other LLM selection algorithms.

1 INTRODUCTION

Large Language Models (LLMs) have transformed numerous applications with their ability to summarize, generate, and interpret text. Due to different underlying training configurations and model structures, LLMs can show a wide variation in terms of their performance for different tasks, raising the need to identify the best-performing model for a given task. However, with the sudden increase in the number of LLMs available and the complexity of tasks they are asked to perform, this challenge has become increasingly complex (Shnitzer et al., 2023). Indeed, some LLM operators even offer users the opportunity to build customized LLM agents, e.g., OpenAI's marketplace or Azure's Assistants (Microsoft, 2025), which may greatly expand the set of agents available to complete tasks. A natural question is then: **How should we select the best LLM agent to complete a given task?**

Selecting the most suitable LLM for a specific task or sequence of subtasks presents significant challenges, particularly in terms of computational efficiency and performance optimization, as has been explored in prior work (Zhang et al., 2024; Xia et al., 2024; Zhao et al., 2023). Due to computational, monetary, and latency concerns, simply running a query or task through all the available models and choosing the model that yields the best performance for that query is not feasible (Shazeer et al., 2017), while simply selecting the largest LLM for every task may be either prohibitively expensive (as these LLMs tend to be the most costly) or ignore the potential of highly specialized LLMs to perform well on specific types of tasks. These naïve approaches become particularly infeasible for **tasks that are too complicated or difficult for an LLM to handle alone**.

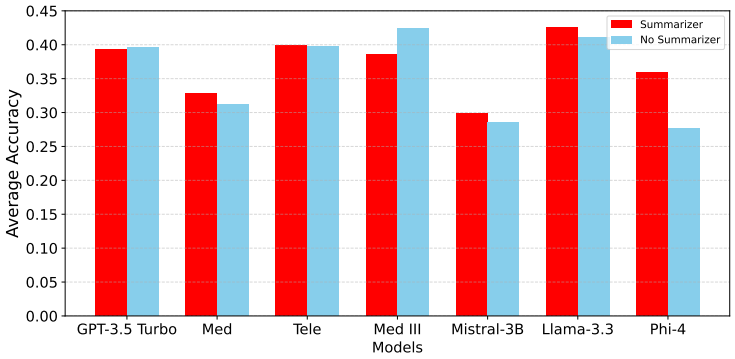

Figure 1: Average accuracies of LLMs (Med: GPT 4o finetuned on medical data, Tele: GPT 4o finetuned on telecommunications questions, Med III: GPT 4o finetuned on MIMIC III) on the medical diagnosis prediction task when the summarizer is chosen uniformly at random. Including a summarizer improves the average performance of most models, showing the value of breaking down a larger task into a pipeline of smaller ones. More detailed explanations about the models, experimental setup and results for this experiment are in Sec. 5.

These complicated tasks are generally broken down into multiple subtasks, each of which can then be completed by a different LLM, resulting in an exponential growth of possible LLM combinations, each of which will introduce a different success rate and monetary cost. Indeed, such use cases for LLM agents are only becoming more popular, with the rise of "agentic" AI for decomposing and completing complex, multi-step tasks (Qiu et al., 2024b). For example, Figure 1 shows the average accuracy of seven LLMs on the task of predicting a medical diagnosis from medical reports. We see that using a summarizer LLM first and then passing its output as the prompt to a diagnoser LLM, i.e., *using a pipeline approach, performs better than using a single LLM* for this task. Only the Med III model sees a visible drop in accuracy from using a summarizer in this task, which can be attributed to the fact that Med III is a finetuned GPT-4o model from the same dataset. *Using a summarizer also changes the optimal model choice for the diagnosis task*: while the Med III model performs best across all LLMs without summarizers, Llama performs best when a summarizer is used. Thus, the LLM selected for an earlier task in the pipeline may affect the optimal choice later in the pipeline, complicating decision-making. Similarly, LLMs selected for the summarization task can influence the *cost* of using different LLMs for the diagnosis task: the diagnoser cost will generally be proportional to the number of input tokens, which depends on the length of the summary provided by the chosen summarizer LLM. To the best of our knowledge, **ours is the first work to examine cost-effective LLM selection within such a pipeline of tasks**.

Research Challenges. In practice, users may not have historical data on LLM performance for specific types of tasks, especially if some candidate LLMs are customized "assistants." Thus, a natural solution is to take an *online approach*, where the user both learns and optimizes the LLM selection as it sends queries/tasks. Online LLM selection for a specific task can be viewed as an instance of the classical contextual multi-armed bandit (MAB) problem (Chu et al., 2011; Chen et al., 2013). Each LLM is modeled as an "arm," and the MAB algorithm sequentially pulls the arm (i.e., submits queries to the LLM), monitoring the query success and adapting future LLM selections accordingly. MAB balances exploration, i.e., trying new models in the hope that they outperform the identified current best model; with exploitation, i.e., selecting the best model found so far. Context, i.e., an embedding of the current query, is used to predict the success of each LLM on this query.

Conventional contextual MAB methods do not capture the *sequential* nature of LLM selection in our pipeline-of-tasks scenario: we must select one LLM for *each* subtask. Combinatorial variants of contextual MAB (Chen et al., 2018), in which a combination of arms (here, LLMs) are chosen in each round, is closer to our scenario, but it requires selecting LLMs for each task at once, instead of selecting the next LLM in the pipeline after observing the results from prior LLMs. Moreover, it is not clear how to modify these algorithms to account for both the success rate and the monetary cost of using each LLM. Thus, we propose a novel MAB variant designed specifically for *sequential* LLM selection, which we show outperforms conventional MAB methods.

The main **contributions** of this work are as follows:

- We introduce a novel **problem formulation** of selecting a pipeline of LLMs to solve a task decomposed into interconnected, smaller subtasks.

- We **adapt contextual MAB algorithms** into the Sequential Bandit algorithm, which sequentially selects LLMs in the task pipeline. Sequential Bandits utilizes neural networks to effectively learn each LLM's success at each subtask in the pipeline and optimize a combination of LLM success and cost in an online manner.
- To evaluate SeqBandits, we **create a new diagnosis prediction dataset** from an existing medical dataset (Johnson et al., 2016) of deidentified patients' medical reports.
- We **experimentally show** that SeqBandits identifies better LLMs than existing MAB algorithms for pipelined tasks on medical diagnosis and telecommunications datasets.

We first outline related work (Sec. 2) and describe our problem formulation (Sec. 3). We present our SeqBandits algorithm in Sec. 4 and its experimental evaluation in Sec. 5, then conclude in Sec. 6.

2 Related Work

The proliferation of LLMs has led to growing interest in methods to predict the best-performing LLM. We divide these strategies into budget-aware frameworks, LLM cascades, and LLM routing.

Budget Constrained Online Algorithms and Bandits. These algorithms maximize cumulative reward subject to a hard cap on total resource consumption. Primal–dual schemes embed standard regret minimizers as black-box components to enforce long-term resource constraints via dual variables (Castiglioni et al., 2022), which can extend to integrate budgeted expert-query mechanisms that judiciously allocate a limited number of advice calls to sharpen decisions (Benomar et al., 2024). Non-stationarity and adaptive primal–dual updates have been shown to ensure constraint satisfaction even as cost and reward distributions shift over time (Liu et al., 2022). Most recently, weakly adaptive regret minimizers have been woven into primal–dual frameworks to simultaneously honor strict budget and return-on-investment limits (Castiglioni et al., 2024). Recently, contextual bandit approaches for multi-LLM selection under evolving contexts have been proposed, highlighting the difficulty of adapting model choices online when input distributions shift (Poon et al., 2025). None of these, however, use sequential decision frameworks, as proposed in our setting.

Cost-Efficient LLM Cascades. In a typical cascading framework, inputs are processed through a pre-determined sequence of LLMs, from the least to the most resource-intensive. At each stage, the system evaluates the output to determine whether to accept the result or continue to the next model in the sequence (Zhang et al., 2024). Recent advancements have focused on integrating cascading with routing strategies, i.e., routing a query to the "best" LLM (Chen et al., 2023; Dekoninck et al., 2025b). Approaches like the Mixture of Thought representations combine chain-of-thought and program-of-thought prompts, in order to adaptively route simpler queries to smaller, less costly models, reserving more complex tasks for larger models (Cheng et al., 2023; Gao et al., 2023). This strategy has demonstrated significant reductions in inference costs while maintaining accuracy comparable to using the most robust LLM alone (Yue et al., 2024). However, cascading can be inefficient if the sequence is not optimally configured, as each input may need to pass through multiple models before reaching an adequate response (Dekoninck et al., 2025a). Unlike these cascading frameworks, Sequential Bandits considers a pipeline of LLMs, in which the different LLMs perform different subtasks that feed into each other (Li, 2025).

Model Selection and Adaptive Routing. Dynamic routing mechanisms, which intelligently direct queries to the most appropriate LLMs, can significantly improve performance and computational resources (Varangot-Reille et al., 2025; Somerstep et al., 2025), as unlike cascades, they do not make multiple passes through different LLMs. Systems like Tryage propose context-aware routing mechanisms that optimally select expert models based on individual input prompts (Hari & Thomson, 2023). This approach allows users to explore trade-offs between task accuracy and secondary goals like minimizing model size and improving response readability (Sikeridis et al., 2024). Other approaches like Zooter train a routing function using reward models' scores as supervision signals (Lu et al., 2023), efficiently directing queries to specialized LLMs (Chen et al., 2025).

Within the popular mixture-of-experts framework, Routing Experts introduces a dynamic expert scheme for multimodal LLMs that also aims to learn more efficient inference pathways (Wu et al., 2025; Saha et al., 2024; Liu et al., 2024). Other works focus on the challenge of dynamic routing, where new, previously unobserved LLMs become available at test time. These strategies generalize by representing each LLM as a feature vector (Jitkrittum et al., 2025). Frameworks like

AutoMix (Aggarwal et al., 2025) focus on predicting LLM success, e.g., with a few-shot self-verification mechanism to estimate output reliability from smaller LLMs (Ding et al., 2024). Recent work like MixLLM and BEST-Route has expanded routing by introducing architectures that dynamically allocate queries across specialized LLMs, optimize test-time compute for accuracy–latency trade-offs, and adapt routing policies to real-time budget constraints (Ding et al., 2025; Wang et al., 2025). These methods, however, focus on selecting a single LLM per query, and cannot be directly applied to selecting a pipeline of LLMs as in our setting.

3 Problem Formulation

We consider multiple rounds $t = 1, 2, \ldots$, where each round is defined by an incoming query q_t. In our formulation, we assume that the breakdown of a task into simpler subtasks $\{T_1, T_2, ..., T_k\}$ is given. These subtasks form a directed acyclic graph (DAG), as the output of a subtask becomes the input of the next task in the pipeline. For example, a query could be a medical diagnosis prediction based on a provided medical report, broken into the subtasks of (i) summarizing the report, (ii) validating the summary, and (ii) predicting a diagnosis given the validated summary. In the remainder of the paper, our aim is to select the LLM returning the highest accuracy for each subtask. Note that a special case of this setting is a query with one subtask (itself). Figure 2 illustrates our pipelined/sequential problem setting. This approach is in essence similar to the popular chain-of-thought (CoT) prompting (Wei et al., 2023), but we *allow different LLMs to complete each subtask*, unlike CoT in which a single LLM decomposes the task and completes each subtask. We formalize this selection problem below. Throughout, we denote by $[N]$, $N \in \mathbb{Z}^+$, the set $\{1, 2, \ldots, N\}$.

Bandit Formulation. In this paper, we model the LLM selection problem as a neural contextual bandit. We consider T rounds. The subtasks $\{T_1, T_2, ..., T_k\}$ correspondingly have $\{N_1, N_2, ..., N_k\}$ available LLMs, each of which we refer to as an "arm" for a particular subtask. We use $\mathcal{M}$ to denote the set of all LLMs. In every round t, the agent selects an arm (LLM) for each of the subtasks corresponding to query q_t, and we name the overall set of arms selected as the **super arm** S_t, following the nomenclature of the combinatorial bandit literature (Chen et al., 2013). Formally, for each subtask T_i where $i \in \{1, \ldots, k\}$, select an arm $a_{i,j}$ where $j \in [N_i]$. Then $S_t = (a_{1,j}, a_{2,j}, \ldots, a_{k,j})$ where $a_{i,j}$ represents the arm j selected from $[N_i]$ for subtask i. When a super arm S_t is selected in round t, the agent observes the **base arm rewards** of the chosen super arm, namely $\{r_{t,j}\}_{a_{i,j} \in S_t}$ and receives a **total (super arm) reward** of $R(S_t, \mathbf{r}_t)$ where $\mathbf{r}_t = [r_{t,j}]_{a_{i,j} \in S_t}$, i.e., the super arm reward is a function of the individual base arm rewards. In our setting, the base arm rewards correspond to the "goodness" of the result for the subtask to which the arm belongs, and the super arm reward is a measure of goodness of the LLMs' collective result on the query, which is a function of the selected base arm rewards. For example, the super-arm reward could be the accuracy of the output returned by the last LLM in the pipeline for a prediction task like medical diagnosis, which is the final output returned to the user. We also account for the **cost** associated with deploying the selected LLM for each subtask. While we take this to be the monetary cost, it could also be taken as the energy consumed or latency of the LLM's inference on the given subtask. We calculate the cost in terms of the number of input and output tokens of the chosen LLM for the given subtask and denote as $C_j(q_t)$ our predicted cost of using LLM j for a subtask's query q_t. We give more detail about how C_j is constructed in the next section.

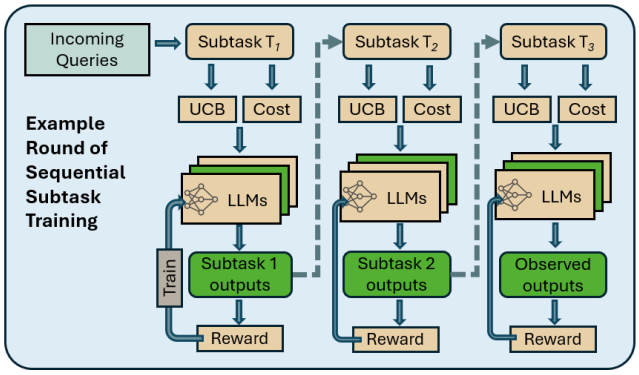

Figure 2: Our Sequential Bandits problem setting and approach, where an incoming query is split into simpler subtasks in a sequential pipeline. An LLM is chosen to complete each subtask and the output of an LLM is fed as the input to the next subtask's LLM in the pipeline.

Context, Reward and Cost Estimation. In every round t, the agent observes the **context** $\mathbf{x}_t(a_{i,j})$ of arm $a_{i,j}$ for each $i \in \{1, \ldots, k\}, j \in [N_i]$, which helps guide the prediction of LLM j's success on subtask i. The context $\mathbf{x}_t(a_{i,j})$ in our application is a function of the given prompt for subtask $i \in \{1, \ldots, k\}$, which is the output of the LLM chosen for the previous subtask $i-1$ as shown in Figure 2, as well as features of the LLM corresponding to arm (i, j). In our experiments, these features include a description of the LLM's capabilities, as well as whether it was finetuned on a specific dataset for a certain task or if it is just a general-use LLM that is not finetuned. We will give more details on how we construct the context of the arms in the Supplementary Material. We denote the description of an LLM j as d_j and the overall set of descriptions of all available LLMs as $\mathcal{D}$.

Following prior works on neural bandits, we use a neural network(s) to learn the underlying reward function(s). We assume that $\forall t \in [T]$, the base arm reward $r_t(a_{i,j})$ of arm $a_{i,j}$ (i.e., LLM j on subtask i) is generated as follows (Zhou et al., 2020; Zhang et al., 2021):

$$r_t(a_{i,j}) = h_{i,j}(\mathbf{x}_t(a_{i,j})) + \xi_t \tag{1}$$

where $h_{i,j}$ is the underlying unknown reward function for arm $a_{i,j}$ and ξ_t is zero-mean noise, modeling uncertainty in the "goodness" of the LLM output. We allow separate reward functions for the different (subtask, arm) combinations, since each combination corresponds to a different model structure and task. To learn the base arm reward function $h_{i,j}$, we use a fully connected neural network with depth $L+1 \geq 3$ and width n (Zhou et al., 2020; Zhang et al., 2021):

$$f_{i,j}(\mathbf{x}; \boldsymbol{\theta}) = \sqrt{m}\mathbf{W}_{i,j}^{(L)}\sigma(\mathbf{W}_{i,j}^{(L-1)}\sigma(...\sigma(\mathbf{W}_{i,j}^{(0)}\mathbf{x})) \tag{2}$$

where $\sigma(x) = \max\{x, 0\}$ is the ReLU activation function and $\boldsymbol{\theta}_{i,j} = [\text{vec}(\mathbf{W}_{i,j}^{(0)})^T, ..., \text{vec}(\mathbf{W}_{i,j}^{(L)})^T]^T$ is the weight vector. We denote the weight vector in round t for arm $a_{i,j}$ as $\boldsymbol{\theta}_{i,j}^t$. We let the gradient of the neural network be $\mathbf{g}_{i,j}(\mathbf{x}; \boldsymbol{\theta}_{i,j}) = \nabla_{\boldsymbol{\theta}_{i,j}} f_{i,j}(\mathbf{x}; \boldsymbol{\theta}_{i,j})$.

The cost $C_j(q_t)$ for LLM j is modelled as the number of input and output tokens, multiplied by LLM j's corresponding per input token and output token costs in Microsoft Azure (see the Supplementary Material for more details). We know the number of input tokens since q_t is given. Since we do not know the number of output tokens LLM j will output in advance for a given prompt, we train an output token length prediction model as in Qiu et al. (2024a), as detailed in the Experiments section.

Agent Objective. The main objective of the agent in this setting is to maximize the reward (accuracy) while also minimizing the cost of deploying LLMs for the tasks. To combine these two metrics into a single one, we define the **net reward** at time t as $N(t) = R(S_t, \mathbf{r}_t) - \boldsymbol{\alpha} \cdot \mathbf{C}(S_t)$ where $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_k]$ trades off the relative importance of reward and cost for each subtask and $\mathbf{C}(S_t)$ is the vector of costs associated with the chosen LLMs. Maximization of reward also leads to the minimization of **regret**, a standard MAB metric that measures the gap in reward between the optimal and selected arms. More formally, the regret $\mathcal{R}(T) = \sum_{t=1}^{T}(R(S_t^*, \mathbf{r}_t^*) - R(S_t, \mathbf{r}_t))$, where S_t^* represents the optimal super arm at round t, which is the combination of arms that yield the highest reward, and $R(S_t, \mathbf{r}_t)$ is the reward of our chosen super arm at round t.

4 Sequential Bandits Algorithm

In this section, we present our proposed algorithm, Sequential Bandits (Algorithm 1), which aims to maximize the net reward. Sequential Bandits is a neural network based contextual bandit algorithm that initializes a neural network for every (subtask, LLM) combination. Figure 2 illustrates how these networks are trained using reward feedback, which is formally described in Algorithm 1.

The task is first divided into simpler subtasks. For the first subtask, for every available LLM, we construct an upper confidence bound (UCB) on the reward using the neural network estimate for exploitation and its gradient for exploration (line 6), similar to Zhou et al. (2020). Here we denote the ℓ_2 norm by a positive definite matrix $\mathbf{A}$ by $\|\mathbf{x}\|_{\mathbf{A}} := \sqrt{\mathbf{x}^T\mathbf{A}\mathbf{x}}$. However, unlike Zhou et al. (2020), our algorithm includes a cost sensitivity parameter α_i, which is multiplied by the cost term C_j, the predicted cost of using the chosen LLM j for subtask i, and then subtracted from the UCB term. Setting $\alpha_i = 0$ reduces to the cost-agnostic setting. Since the cost term is subtracted from the UCB term in line 6, this leads to a tradeoff between the accuracy and cost. As the value of α_i increases, the algorithm will prefer cheaper models, prioritizing cost over accuracy for task i.

As described in Sec. 3, the input to the neural network includes embeddings of the description d_j of each LLM j, as well as the incoming query q_t. We then choose the LLM that has the highest reward estimate (line 7) and pass the output of the chosen LLM as the input prompt for the next subtask in the pipeline (lines 8-9). For the other subtasks, we follow the same steps, except that instead of the input query q_t, the input prompt for the LLMs in the next stage of the pipeline (next subtask) is the output from the previously chosen LLM (lines 11-14). After choosing an LLM for every subtask, we observe their corresponding rewards and the overall super arm reward $R(S_t, \mathbf{r}_t)$, a function of the subtask rewards (line 17). We then update the weights of the chosen LLMs' corresponding networks using the observed rewards, as well as the exploration parameter (lines 18-19).

Incorporating Costs. An alternative method of incorporating the cost term would impose a cost constraint (e.g., a monetary budget per query or task) instead of incorporating it into the objective. However, *LLM inference costs are inherently uncertain before query execution*, due to their dependence on the number of output tokens. Thus, a hard budget constraint could favor subtasks early in the pipeline, as underestimations of their costs could then lead to severe budget constraints for later subtasks. Incorporating cost into our objective also allows us to tune α_i for different subtasks i, which may have different inherent costs: for example a summarization task may be quite expensive as its input consists of a long text to be summarized.

Differences from Existing Neural MAB Algorithms. Prior neural contextual bandit algorithms (Zhou et al., 2020; Xu et al., 2020) can be naïvely adapted to our problem setting by simply using them for each subtask's LLM selection in sequence, which we use as baselines in Sec. 5. However, such algorithms train and use *one* neural network for reward estimation and LLM selection for each subtask. Using this single model for all LLMs does not account for the fact that different models imply a different inherent reward function for the different subtasks for each LLM. Thus, Sequential Bandits uses a *separate* neural network for every (subtask, arm) combination. Experimentally, we find that *using a separate neural network encourages more exploration*, as using the same neural network estimator for each (subtask, arm) combination leads to more similar success estimates. The relative ranking of the different arms is then dominated by the (deterministic) cost estimates, limiting exploration and increasing regret. We note that *using multiple neural networks does not increase our training overhead*: while we use different networks for every (subtask, LLM) combination, in each round we only train the neural networks of the LLMs that we have chosen for every subtask. Thus, our required compute is no greater than the other neural contextual MAB algorithms.

5 Experiments

In this section, we present the results of our proposed Sequential Bandits algorithm on two use cases: medical diagnosis prediction and telecommunications question answering tasks. We compare our algorithm with the following baselines: (1) **Random**, which randomly selects an LLM for each subtask; (2) **Llama** or **Tele**, which always selects Llama or Tele for the given subtasks (Llama is chosen as it is the best performing model across tasks for the medical setting and Tele is the best for the telecom setting); (3) **Cost-Aware NeuralUCB** (Zhou et al., 2020), which is the cost sensitive version of NeuralUCB that uses a neural network for each subtask's reward prediction and adds the same weighted cost term in the objective as used in Sequential Bandits; and (4) **Cost-Aware NeuralLinUCB** (Xu et al., 2020), the cost-sensitive version of NeuralLinUCB, which makes use of neural networks for each subtask to learn representations of the contexts, and applies a linear model on these learned features to predict the rewards, then adds the weighted cost term to the objective, and (5) **Cost-Aware NeuralUCB Joint**, which makes use of neural networks for each subtask as (3) but selects LLMs for subtasks all at once rather than sequentially. Comparisons to these algorithms respectively demonstrate the value of (1) having a non-static algorithm that learns over time, (2) intelligently selecting a LLM for each subtask, (3,4) using a separate neural network to predict each LLM's performance on each subtask, and (5) sequentially selecting LLMs.

5.1 Experiment Settings

We evaluate Sequential Bandits on two **datasets**: one created from MIMIC-III (see the next subsection), a comprehensive clinical database containing de-identified health-related data from over 40,000 critical care patients (Johnson et al., 2016); and TeleQnA, a dataset comprising 10,000 multiple-choice questions designed to assess LLMs' knowledge in telecommunications (Maatouk

Algorithm 1 Sequential Bandits

Input: LLMs $m \in \mathcal{M}$, descriptions $d \in D$, queries $[q_1, q_2, \ldots, q_T]$, number of gradient descent steps J, learning rate η, cost weight parameters α_i, number of rounds T, regularization parameter λ, initialize $\mathbf{Z}_0(a_{i,j}) = \lambda\mathbf{I}$ for all arms $a_{i,j}$.
for $t = 1, \ldots, T$ **do**
  Observe descriptions $d \in D$ and subtasks $i \in \{1, 2, ..., |\mathcal{T}|\}$
  for subtask $i = 1, \ldots, |\mathcal{T}|$ **do**
    if $i = 1$: **then**
      Compute $\forall j \in [N_i],\ u_{i,j} = f_{i,j}(q_t, d_j) + ||\mathbf{g}_{i,j}(\mathbf{x}_t(a_{i,j}); \boldsymbol{\theta}_{i,j}^{t-1})/\sqrt{n}||_{\mathbf{Z}_{t-1}^{-1}(a_{i,j})} - \alpha_i C_j(q_t)$
      $s_i = \text{argmax}_j(\mathbf{u}_i)$ (LLM chosen for subtask i)
      Pass query q_t through LLM s_i
      Observe output p_{i+1} of the chosen LLM
    else
      Compute $\forall j \in [N_i],\ u_{i,j} = f_{i,j}(p_i, d_j) + ||\mathbf{g}_{i,j}(\mathbf{x}_t(a_{i,j}); \boldsymbol{\theta}_{i,j}^{t-1})/\sqrt{n}||_{\mathbf{Z}_{t-1}^{-1}(a_{i,j})} - \alpha_i C_j(p_i)$
      $s_i = \text{argmax}_j(\mathbf{u}_i)$ (LLM chosen for subtask i)
      Pass prompt p_i through LLM s_i
      Observe output p_{i+1} of the chosen LLM
    end if
  end for
  Play super arm S_t and observe rewards $\{r_{t,j}\}_{a_{i,j} \in S_t}$ and super arm reward $R(S_t, \mathbf{r}_t)$
  For selected arms $a_{i,j} \in S_t$, update $\mathbf{Z}_t(a_{i,j}) = \mathbf{Z}_{t-1}(a_{i,j}) + \sum_{a_{i,j} \in S_t} \mathbf{g}_{i,j}(\mathbf{x}_t(a_{i,j}); \boldsymbol{\theta}_{i,j}^{t-1}) \mathbf{g}_{i,j}(\mathbf{x}_t(a_{i,j}); \boldsymbol{\theta}_{i,j}^{t-1})^T / n$
  Update weights of selected arms $\boldsymbol{\theta}_{i,j}^t$ by minimizing the MSE loss using gradient descent with step size η for J iterations by using the reward feedback.
end for

et al., 2023). We thus assess our framework's performance across diverse, domain-specific tasks. We use pipelines with 2 and 3 subtasks respectively for the medical and telecommunications datasets.

Subtasks. Our two subtasks on the medical dataset are (1) a summarizer that summarizes the long medical report, whose summary is then fed to (2) a diagnoser that gives a diagnosis based on the summary. The **rewards** for the summarization subtask are obtained using an evaluation LLM that is fed the prompt, context, and benchmark. The reward for the diagnosis subtask given the summarized report compares the diagnoses that the LLM outputs to the actual diagnoses of the patient, which also serves as the super arm reward. For example, if a patient has two diagnoses, and the second subtask's LLM correctly predicts one of the two diagnoses of the patient, we assign a reward of 0.5.

For TeleQnA, we have a 3 subtask structure. We start with a summary subtask with a similar reward metric as the medical dataset's summary task. The second subtask is to answer the question, for which we compute the LLM rewards by comparing their output choice (among the 4-5 options) to the correct one. The third subtask was to explain the answer obtained from the previous subtask, whose reward is obtained by comparison to TeleQnA's explanation benchmark.

Cost Prediction. For the output token length prediction model, which is used to construct the expected cost (C_j in Algorithm 1) of LLM query execution, we train a Bert regression model on the LMSYS-Chat-1M dataset (Zheng et al., 2023) using L1 loss, following Qiu et al. (2024a). This model is kept fixed during the online training loop. We provide more information about the training of this model and the loss curves in the supplementary material.

All of the **models** we used in our experiments were deployed on Microsoft Azure. The models include a combination of base models (GPT-3.5-turbo, GPT-4o, Llama-3.3-70B-instruct, Mistral-3B, Phi-4), finetuned models that were finetuned using Azure on general medical and telecom knowledge, and GPT-3.5-turbo assistants that used file search prompted with relevant field knowledge. The models were selected in a way such that there are low performing/SLMs, base models, and finetuned/custom domain knowledge possessing LLMs (Figure 1). However, as the LMSYS-Chat-1M dataset (Zheng et al., 2023), which we use to train our token length prediction model, does not con-

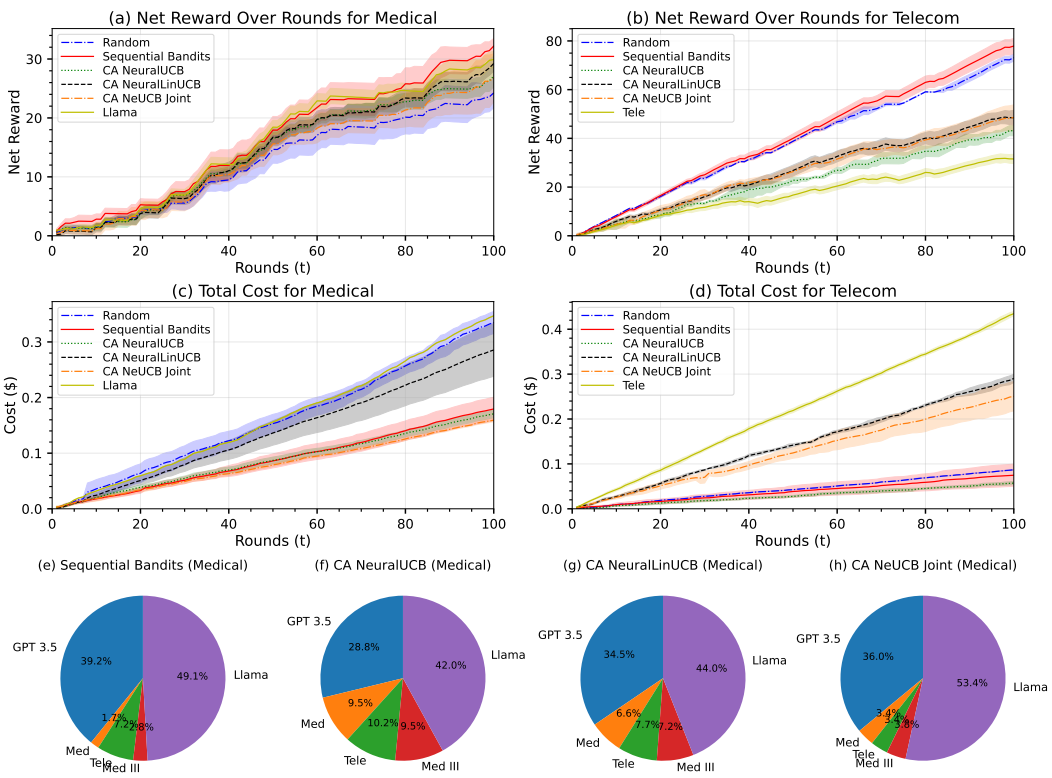

Figure 3: (a) Net reward of our algorithm compared with other baselines for the medical pipeline (b) Net reward of our algorithm compared with other baselines for the telecom pipeline (c) Total cost incurred by the algorithms for the LLM selections they make for the medical pipeline (d) Total cost incurred by the algorithms for the LLM selections they make for the telecom pipeline (e)-(h) Model selections for the diagnosis subtask in medical experiments

tain the Mistral and Phi models, we omit them in the experiments presented in the main paper but include them in the Supplementary Material's cost-agnostic ($\boldsymbol{\alpha} = \mathbf{0}$) results. The models ordered from cheapest to most expensive per token costs are: GPT-3.5-turbo, Llama 3.3, Med, Tele, Med III.

The n2c2 smoking dataset (Uzuner et al., 2007) was used to fine-tune the Med model, with individual reports constructed into input/label pairs by extracting the diagnoses embedded within the reports. Given the format of the medical reports, they were initially processed through a summarizer to extract general medical knowledge before being utilized to fine-tune the GPT-4o models. The other two fine-tuned models, Med III and Tele, were respectively fine-tuned with the MIMIC-III and TeleQnA datasets. **Evaluations** were performed using the Microsoft Azure evaluator, and the fine-tuned models generally performed better than the base models on the task they were tuned on.

5.2 Diagnosis Prediction Dataset

There are some widely available medical datasets that include medical reports for de-identified patients. However, to our knowledge, there is no available dataset that is specifically tailored towards diagnosis prediction based on medical reports. Hence, we developed our own dataset from MIMIC III (Johnson et al., 2016) to be able to successfully assess Sequential Bandits' performance.

Johnson et al. (2016)'s MIMIC III dataset contains the admission diagnoses as well as diagnoses identified later, along with details of patient stay in their reports. Each patient has a number of reports based on their duration in the hospital, which we combine into a single report for each patient. We remove all explicit mentions of the patient's identified diagnoses, as well as diagnosis-related comments. We include the observations made and the test results of the patients in the

reports. Patients we included in the dataset were mainly diagnosed with diseases related to the heart, kidneys, liver, and brain. Our dataset includes 100 medical reports with corresponding diagnoses. We include a more comprehensive list of the diagnoses of the patients in the Supplementary Material and some example patient reports. The full dataset is available in our released code.

5.3 Experimental Results

We now present the experimental results for our medical diagnosis dataset and the TeleQnA dataset. All algorithms were run ten times and five times respectively for the medical and telecommunication settings, with tuned hyperparameters. The shaded regions in Figure 3 indicate standard deviations.

Net Reward and Cost. We first show that *Sequential Bandits obtains a higher net reward $N(t)$ than baseline algorithms on both pipelined settings*. For the medical setting, we consider the reward to be the reward of the final subtask in the pipeline (i.e., diagnosis accuracy) while for the telecom setting, the reward is the sum of the reward for the explanation and multiple choice answer. Figure 3 shows that our algorithm (solid red lines) achieves the highest net reward and also the second or third lowest cost, in both settings. When comparing the final net reward, Sequential Bandits has a %7.60 improvement over the most competitive baseline (Llama) in the medical setting while displaying a %6.51 improvement over the most competitive baseline (Random) in the telecom setting. In fact, when we look at Figure 1, which displays the diagnosis accuracy for the different models in the medical setting, we would expect Llama to outperform Sequential Bandits, as Llama has the highest accuracy among the available models in this setting. However, the fact that our algorithm outperforms Llama shows that it can learn the complex dependencies between the subtasks, as there may be combinations other than Llama for both summarizer and diagnosis that obtain a higher net reward. For the telecom dataset, it can be seen that the Tele baseline performs the *worst* among all the baselines in terms of net reward, despite the fact that it is fine-tuned on this dataset, due to the fact that it incurs a much higher cost, (more than $\$0.4$) over 100 rounds. Random performs the worst in the medical setting while it surprisingly performs well in the telecom setting.

Analyzing LLM Selections. We finally take a closer look at which LLMs are selected by each algorithm for the medical diagnosis subtask. Figure 3(e) shows that Sequential Bandits selects Llama and GPT 3.5 the most often (%49.1 and %39.2 respectively), which enables it to have high net reward and low cost as these are the cheapest models, and Llama and GPT 3.5 also have the highest and second highest accuracy for this subtask respectively. The other baselines, as shown in Figure 3(f)-(h), select these two models less frequently and make more suboptimal choices such as selecting the Med model more often, which has the lowest accuracy. CA NeUCB Joint selects similar models to our algorithm for diagnosis, but it selects different summarizers from Sequential Bandits which results in the performance difference. The pie charts for the telecommunications and medical settings, indicating the model choices, can be found in the Supplementary Material.

Additional Experiments. We include more experimental results in the Supplementary Material, including (i) the regret achieved by each algorithm and model accuracies on subtasks. We also present results for (ii) a 3 subtask version of the 2 subtask medical pipeline in the main paper, and (iii) cost-agnostic ($\boldsymbol{\alpha} = 0$) versions of the Medical Diagnosis task, as well as a single-subtask TeleQnA task. Since the cost can also be interpreted as response latency, we further present results on the model response latencies, showing they are hard to predict.

6 Conclusion

In this paper, we introduced a novel approach to selecting a pipeline of LLMs for executing decomposed tasks, employing a contextual MAB algorithm that sequentially chooses the best LLM for each sub-task in an online manner. Our approach, Sequential Bandits, leverages neural networks to model the expected success of each LLM, thereby enabling more effective task completion across a sequence of dependent subtasks. We demonstrated the effectiveness of our method through experiments involving medical diagnosis prediction and multiple choice telecommunications question answering. Our work provides a framework that can be adapted to various domains where task decomposition is feasible. Future work includes handling a broader range of tasks with varying degrees of difficulty and interdependencies, and providing regret guarantees or other performance bounds, as well as incorporating the task decomposition itself into the online learning framework.

Reproducibility Statement We have taken several steps to ensure the reproducibility of our work. The full problem formulation, algorithmic details, and pseudocode for Sequential Bandits are provided in the main text (Secs. 3–4) and Algorithm 1. Details about hyperparameter tuning and other experiment settings, e.g., formation of the contexts using embeddings, may be found in the Supplementary Material. All datasets we used are either public datasets mentioned in the experiment settings or available in our code repository. We also release our training and evaluation scripts to reproduce the main results and figures. Evaluation metrics, baseline implementations, and cost estimation methods are fully described in Sec. 5 and the Supplementary Material. Together, these materials provide the necessary resources to reproduce our findings.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models, 2025. URL https://arxiv.org/abs/2310.12963.

Ziyad Benomar, Evgenii Chzhen, Nicolas Schreuder, and Vianney Perchet. Addressing bias in online selection with limited budget of comparisons, 2024. URL https://arxiv.org/abs/2303.09205.

Matteo Castiglioni, Andrea Celli, Alberto Marchesi, Giulia Romano, and Nicola Gatti. A unifying framework for online optimization with long-term constraints, 2022. URL https://arxiv.org/abs/2209.07454.

Matteo Castiglioni, Andrea Celli, and Christian Kroer. Online learning under budget and ROI constraints via weak adaptivity. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5792–5816. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/castiglioni24a.html.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance, 2023. URL https://arxiv.org/abs/2305.05176.

Lixing Chen, Jie Xu, and Zhuo Lu. Contextual combinatorial multi-armed bandits with volatile arms and submodular reward. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 3251–3260, Red Hook, NY, USA, 2018. Curran Associates Inc.

Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit: General framework and applications. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 151–159, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/chen13a.html.

Zhijun Chen, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Dingqi Yang, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on llm ensemble, 2025. URL https://arxiv.org/abs/2502.18036.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lH1PV42cbF.

Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 208–214, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/chu11a.html.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. Polyrating: A cost-effective and bias-aware rating system for llm evaluation, 2025a. URL https://arxiv.org/abs/2409.00696.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for llms, 2025b. URL https://arxiv.org/abs/2410.10347.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=02f3mUtqnM.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks V. S. Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute, 2025. URL https://arxiv.org/abs/2506.22716.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10764–10799. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Surya Narayanan Hari and Matt Thomson. Tryage: Real-time, intelligent routing of user prompts to large language models, 2023. URL https://arxiv.org/abs/2308.11601.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Zifeng Wang, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient llm inference, 2025. URL https://arxiv.org/abs/2502.08773.

Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3:160035, 2016.

Yang Li. Llm bandit: Cost-efficient llm generation via preference-conditioned dynamic routing, 2025. URL https://arxiv.org/abs/2502.02743.

Shang Liu, Jiashuo Jiang, and Xiaocheng Li. Non-stationary bandits with knapsacks. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 16522–16532. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/69469da823348084ca8933368ecbf676-Paper-Conference.pdf.

Zijun Liu, Yanzhe Zhang, Peng Li, Yang Liu, and Diyi Yang. A dynamic llm-powered agent network for task-oriented agent collaboration, 2024. URL https://arxiv.org/abs/2310.02170.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models, 2023. URL https://arxiv.org/abs/2311.08692.

Ali Maatouk, Fadhel Ayed, Nicola Piovesan, Antonio De Domenico, Merouane Debbah, and Zhi-Quan Luo. Teleqna: A benchmark dataset to assess large language models telecommunications knowledge, 2023.

Microsoft. Getting started with azure openai assistants, 2025. URL https://learn.microsoft.com/en-us/azure/ai-services/openai/how-to/assistant.

Manhin Poon, XiangXiang Dai, Xutong Liu, Fang Kong, John C. S. Lui, and Jinhang Zuo. Online multi-llm selection via contextual bandits under unstructured context evolution, 2025. URL https://arxiv.org/abs/2506.17670.

Haoran Qiu, Weichao Mao, Archit Patke, Shengkun Cui, Saurabh Jha, Chen Wang, Hubertus Franke, Zbigniew T Kalbarczyk, Tamer Başar, and Ravishankar K Iyer. Efficient interactive llm serving with proxy model-based sequence length prediction. *arXiv preprint arXiv:2404.08509*, 2024a.

Jianing Qiu, Kyle Lam, Guohao Li, Amish Acharya, Tien Yin Wong, Ara Darzi, Wu Yuan, and Eric J Topol. Llm-based agentic systems in medicine and healthcare. *Nature Machine Intelligence*, 6 (12):1418–1420, 2024b.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. Branch-solve-merge improves large language model evaluation and generation, 2024. URL https://arxiv.org/abs/2310.15123.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, 2017. URL https://arxiv.org/abs/1701.06538.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Dimitrios Sikeridis, Dennis Ramdass, and Pranay Pareek. Pickllm: Context-aware rl-assisted large language model routing, 2024. URL https://arxiv.org/abs/2412.12170.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. CARROT: A cost aware rate optimal router. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025. URL https://openreview.net/forum?id=xEBOy2ze1U.

Özlem Uzuner, Y Juo, and P Szolovits. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association*, 14(5):550–563, 2007. URL http://www.jamia.org/cgi/content/abstract/14/5/550.

Clovis Varangot-Reille, Christophe Bouvard, Antoine Gourru, Mathieu Ciancone, Marion Schaeffer, and François Jacquenet. Doing more with less – implementing routing strategies in large language model-based systems: An extended survey, 2025. URL https://arxiv.org/abs/2502.00409.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. Mixllm: Dynamic routing in mixed large language models, 2025. URL https://arxiv.org/abs/2502.18482.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Qiong Wu, Zhaoxi Ke, Yiyi Zhou, Xiaoshuai Sun, and Rongrong Ji. Routing experts: Learning to route dynamic experts in multi-modal large language models, 2025. URL https://arxiv.org/abs/2407.14093.

Yu Xia, Fang Kong, Tong Yu, Liya Guo, Ryan A Rossi, Sungchul Kim, and Shuai Li. Which llm to play? convergence-aware online model selection with time-increasing bandits. In *Proceedings of the ACM Web Conference 2024*, pp. 4059–4070, 2024.

Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780*, 2020.

Murong Yue, Jie Zhao, Min Zhang, Liang Du, and Ziyu Yao. Large language model cascades with mixture of thoughts representations for cost-efficient reasoning, 2024. URL https://arxiv.org/abs/2310.03094.

Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tkAtoZkcUnm.

Xuechen Zhang, Zijian Huang, Ege Onur Taga, Carlee Joe-Wong, Samet Oymak, and Jiasi Chen. Efficient contextual llm cascades through budget-constrained policy learning, 2024. URL https://arxiv.org/abs/2404.13082.

James Xu Zhao, Yuxi Xie, Kenji Kawaguchi, Junxian He, and Michael Qizhe Xie. Automatic model selection with large language models for reasoning, 2023. URL https://arxiv.org/abs/2305.14333.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with UCB-based exploration. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 11492–11502. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/zhou20a.html.