Let the LLM Stick to Its Strengths: Learning to Route Economical LLM

Yi-Kai Zhang[1,2] **Shiyin Lu**[3] **Qing-Guo Chen**[3] **Weihua Luo**[3]
De-Chuan Zhan[1,2] **Han-Jia Ye**[1,2*]
[1]School of Artificial Intelligence, Nanjing University
[2]National Key Laboratory for Novel Software Technology, Nanjing University
[3]AI Business, Alibaba Group

Abstract

Recently, test-time scaling of Large Language Models (LLMs) has emerged as a practical alternative to parameter and data scaling. Reasoning tasks often require large-scale, RLVR-based LLMs, while more economical LLMs can handle simpler tasks. Routing an LLM tailored to *suitability* (*i.e.*, capability and cost) ensures usability and efficiency. We introduce LLMRec, which routes the most suitable LLM to the user query without pre-inference on the candidate LLM zoo. It pioneeringly reframes the LLM routing problem as a comprehensive recommendation system (RecSys) task. Our core insight is that an LLM's suitability for a query is a complex, latent signal equal to user-item preference. LLMRec systematically engineers features for candidate LLMs (intrinsic attributes and capability distributions), queries (general semantics and meta-dimensional info), and context (inference type, cost budgets). It also incorporates behavioral features to learn high-order interactions. LLMRec is designed to generalize to out-of-domain datasets and adapt to new LLMs as the model zoo evolves. We define the metric with the Pareto frontier under user-specified cost budgets. Across six datasets, LLMRec achieves an average cost reduction of over 38% while maintaining accuracy and consistently outperforming baselines in converging toward the Pareto frontier.

1 Introduction

The rapid growth of large language models (LLMs) [27, 38] has produced a diverse ecosystem of models varying in scale, functionality, and performance. The parameter scaling era gave rise to models ranging from 1B-parameter edge-device variants to 100B+ omni-models [48, 57]. The subsequent test-time scaling era introduced advanced reasoning LLMs like DeepSeek-R1 [15], which leverage reinforcement learning and extended thinking processes for superior performance.

In practice, companies often provide LLMs-as-API-services [19]. For medium-sized business clients, the monthly expenses for these services can reach millions of dollars [1], [53]. A significant challenge arises from this diversity: the capabilities of large or reasoning-heavy LLMs often far exceed the requirements of some downstream tasks, leading to unnecessary costs. Smaller-scale LLMs are typically sufficient for handling simple tasks, but more complex tasks, such as code generation [10] or mathematical reasoning [12], often require techniques like chain-of-thought (CoT), or larger, reasoning LLMs with extended inference tokens [28, 13, 54]. An intuitive idea emerges: can we intelligently route user queries to the most suitable LLM, balancing performance and cost?

A good model routing outcome hinges on cost considerations, primarily driven by two factors [3]: the cost per token and the number of tokens generated. The former reflects the expense of producing

*Corresponding author, email: yehj@lamda.nju.edu.cn.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

a single token, typically scaling with the size of the candidate LLM. The latter is closely tied to the inference mode, or whether the LLM is RLVR-based. The total cost can be defined as the product of these two. A general observation is that larger LLMs or long-chain inference increase costs but improve task completion and accuracy. From that, our model routing goal is to minimize costs without sacrificing accuracy, approaching the Pareto optimum in the trade-off between cost and accuracy.

In this paper, we propose LLMRec, which pioneeringly reframes the model routing problem as a comprehensive recommendation system (RecSys) task. We find the underlying logic of these two problems is equivalent: (**1**) **RecSys** learns the complex match between a user and an item to predict a personalized *Preference*. (**2**) **Model Routing** learns the optimal match between a query and an LLM to evaluate its comprehensive *Suitability* (*i.e.*, capability and cost). Importantly, both tasks share an assumption: RecSys is effective because a user's preference is a complex, latent signal. Similarly, an LLM's suitability for a query is a profound, implicit match determined by its specific capabilities, domain expertise, and cost, which can be learned from massive historical performance logs.

Specifically, we introduce the feature engineering paradigm from RecSys into model routing. In LLMRec, *queries* and *LLMs* correspond to *users* and *items*. We systematically construct features for: (**1**) Model and Query: An LLM representation captures intrinsic attributes (like architecture, scale, origin, training details, *etc.*) and capability distributions (as performance on benchmarks at release and on a core set we constructed). For query representation, we consider general semantic embeddings and meta-dimensional information (including high-level evaluations like answer difficulty, reasoning level, and domain category). (**2**) Context: As RecSys dynamically adjusts recommendations based on different times or locations, we use the user-specified infer type (*e.g.*, self-consistency, CoT, and Tree-of-Thought) and cost threshold as contextual features, enabling adaptation to different service-level objectives. (**3**) Behavioral sequences: Furthermore, we build dynamic interaction features based on the performance of candidate LLMs on **i**) a core set of tasks and **ii**) on top-k nearest training neighbors of the current query. Like modern RecSys, the core of the LLMRec framework learns the complex, high-order cross-information among model, query, behavioral, and contextual features. This drives it to accurately predict a *suitability* score for each candidate LLM, recommending the one that best approaches the Pareto optimum for the given cost constraint.

LLMRec is designed to be robust, generalizing to out-of-domain datasets, and adapting to new LLMs as the model zoo evolves. While maintaining accuracy, LLMRec reduces costs by an average of over 38% across six datasets. It consistently outperforms all baselines in converging toward the Pareto frontier under various cost budgets. Our main contributions are as follows:

- We are the first to systematically apply recommendation systems to model routing, achieving economical, efficient, and iterative optimization.
- We implemented model, query, behavioral, and contextual features, allowing users to specify the inference type and a cost threshold to constrain the routing.
- We develop a cost-budget-based Pareto metric tailored for LLM routing applications.
- We construct a large-scale training set and routing benchmark, including evaluations on unseen datasets and dynamic model zoos.

LLMRec is versatile, efficient, and adaptable to the evolving LLM landscape, offering a scalable solution for real-world LLM API deployment.

2 Preliminary

We start by presenting the model, query, and pipeline of the LLM router, and discuss its connection to RecSys. We then introduce the deployment and related work of model routing.

2.1 Notations

Key Elements in LLM Routing Consider a scenario where, when using an LLM API, the system provides a candidate LLM zoo $\mathcal{M} = (f^1, f^2, ..., f^M)$. In this case, the user provides a task consisting of an instruction set $\mathcal{D}_{\text{test}} = \{(\mathbf{x}_i, \mathbf{a}_i)\}_{i=1}^N$, where the LLM f^m produces the output as $\mathbf{o}_i = f^m(\mathbf{x}_i; \mathbf{I}_j)$ on input $\mathbf{x}_i$ using inference mode $\mathbf{I}_j \in \mathcal{I}$, and the correct answer is $\mathbf{a}_i$. The accuracy is given by $\text{Acc}(\mathbf{o}_i, \mathbf{a}_i)$. In this paper, we focus on the case that f^m represents a decoder-only text generation LLM. As established in the section 1, the core challenge of LLM routing is to predict the *suitability* of an LLM $f^m \in \mathcal{M}$ for a given query $(\mathbf{x}_i; \mathbf{I}_j)$ *without* first generating the actual output $\mathbf{o}_i$

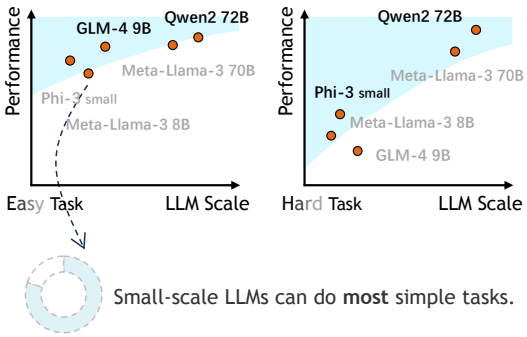


Figure 1: **Comparison of the performance laws** of LLMs with different scales, architectures, and tasks. For most LLM architectures, smaller-scale LLMs can accomplish most simple tasks efficiently; routing to a smaller-scale LLM can reduce deployment costs.

from each candidate model. This *suitability* is conceptually consistent with the foundation of RecSys, which operates by predicting the complex user *preference*.

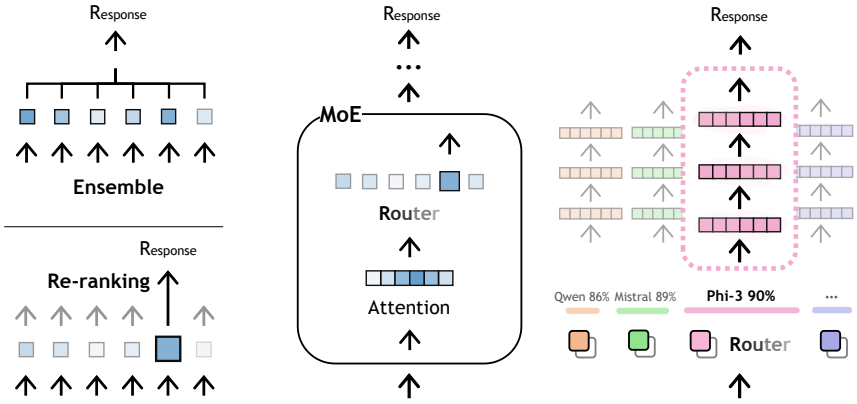
Brief Background on Recommendation System The core task of RecSys is to learn the complex matching relationships between *users* and *items*. A typical process begins by collecting user and item features, their historical interaction logs, and contextual information. The model then learns a user's preference for different items under various scenarios. The ranking task in RecSys estimates the probability of a user clicking on each item.

Cost-effective Routing Pipeline Our task is to select an LLM f^m for each instruction $\mathbf{x}_i$ and the inference mode $\mathbf{I}_j$ to minimize cost while maintaining accuracy. We define the per-token cost of an LLM as t^m, and the goal is:

$$\boldsymbol{f}^* = \underset{f^m \in \mathcal{M}, \mathbf{I}_j \in \mathcal{I}}{\arg\max}\ \mathbb{E}_i \left[\mathrm{Acc}\left(f^m\left(\mathbf{x}_i; \mathbf{I}_j\right), \mathbf{a}_i \right) \right], \quad \text{s.t.} \sum_{i=1}^{n} t^m \cdot |\mathbf{o}_i| \leqslant \epsilon\ , \tag{1}$$

where $\mathbf{o}_i = f^m\left(\mathbf{x}_i; \mathbf{I}_j\right)$. The cost of the input sequence is omitted (as it is fixed for all methods). ϵ represents the user-specified cost threshold. In our formulation, exceeding the cost threshold ϵ incurs some penalty, but it is less critical than losing accuracy. The upper bound of LLM routing performance depends on the number of instructions for which no available LLM can produce a correct answer. In practice, directly optimizing Equation 1 as a hard-constraint problem is intractable. The budget ϵ is a *global* constraint summed over the entire dataset, whereas the router must make a *local* decision for each query $\mathbf{x}_i$. Furthermore, a router trained to satisfy a fixed ϵ is inflexible; it cannot adapt to a different user-specified budget without being retrained. To overcome this, we reframe the problem by treating the cost threshold ϵ as a dynamic input feature. This converts the hard constraint into a learnable condition. The router learns a policy that associates different budget levels with corresponding routing strategies, enabling it to dynamically balance cost and accuracy based on the user's needs. We formally define the router, f^{router}, as a policy that selects a model $f^m \in \mathcal{M}$ given an input $\mathbf{x}_i$, an inference mode $\mathbf{I}_j$, and the cost threshold ϵ.

Definition of the Pareto Front based on Cost Intervals In practical applications, service providers tend to prioritize accuracy as a key performance indicator over cost. This preference stems from their desire for a predictable performance outcome rather than an uncertain level of accuracy after incurring expenses. Therefore, when LLM API service providers implement LLM routing, they often address the dual problem of the optimization presented in the above Equation. This dual problem aims to minimize costs subject to a constraint on a target accuracy. At this point, we introduce the concept of Pareto dominance. For any two routing solutions, say solution f_a^{router} and solution f_b^{router}, characterized by their respective metrics (*e.g.*, accuracy and cost), we state that solution a Pareto dominates solution b (denoted as $f_a^{\text{router}} \succ f_b^{\text{router}}$) if and only if solution a is strictly better than solution b in at least one objective and not worse in the other objectives. If one solution has lower accuracy but also a lower cost than another, neither solution dominates. They may belong to the same



(a) Methods w/ Inference All Candidate LLMs

(b) Mixture-of-Experts

(c) LLM Routing (Ours)

Figure 2: **Comparison of Different Methods for Utilizing pre-trained LLM Libraries**: Ensemble and Re-ranking methods require inference from all candidate LLMs. The Ensemble approach combines the outputs of multiple LLMs, while the Re-ranking selects the most suitable response from all generated outputs. Mixture-of-Experts (MoE) integrates routing within the Transformer layer structure. LLM Routing in our setup directs instruction to one of the candidate LLMs without requiring inference on the target instruction for all LLMs.

Pareto front. If both solutions fall within the user's acceptable range for both cost and accuracy, they are both considered comparatively optimal.

2.2 Related Work

Leveraging the formalized setting, we revisit key features from the section 1:

- **No pre-inference model interactions.** Some model ensemble or cascade strategies aim to select and synthesize the optimal response from all candidate models, considering input-response relationships. Some scoring strategies have been applied in reinforcement learning-based LLM training. Others include classical machine learning techniques [7, 40, 20, 21] and multi-perspective deep ensemble methods [35, 22]. However, these methods can introduce significant delays in LLM APIs, especially with large-scale candidates. In LLMRec, inferring each model for every target instruction is not feasible.
- **Generalization in new scenarios.** LLMRec is designed for LLM APIs with a focus on router scalability. While some transfer learning approaches [5, 29, 59, 39] use proxy source-target metrics based on label distribution matching, they are limited by the target set seen during training [9]. On the other hand, LLM APIs have a flexible natural language output space, meaning the router should generalize zero-shot to unseen user instructions.
- **Extension on updated candidate models.** Some application-specific router frameworks [44, 3, 14] have addressed the issues mentioned above. However, most approaches fix the candidate model zoo in order to stabilize routing training and deployment scenario [36, 16, 17, 46, 37]. Additionally, some mixture-of-experts (MoE) [45, 18, 31] use MLPs within transformer blocks as experts, embedding a router to select among them to reduce inference costs. However, these methods tightly couple router and model parameters, so when the expert zoo is updated, the router needs to adapt through complex incremental learning [41, 64], which may introduce issues like hyperparameter sensitivity and catastrophic forgetting.

In summary, while existing approaches have made significant strides in specific aspects of model routing, LLMRec distinguishes itself by offering a solution that holistically addresses the challenges of pre-inference overhead, zero-shot generalization to new instructions, seamless extension to updated model libraries, and inherent cost-efficiency. Its novel use of a recommendation system framework, as depicted in Figure 3, based on learnable representations of models and queries, allows for dynamic and efficient routing in large-scale LLM API environments. LLMRec introduces a scalable approach

by creating a universal, learnable model representation, turning LLM routing into learning both model and query embeddings. The model representation encodes capabilities and behaviors and is optimized with a dynamic embedding vocabulary. This universality allows new LLMs to quickly index into the embeddings after lightweight inference, unlike the random initialization and re-training needed in Model Spider [63]. Finally, the routing process estimates the relationship between the model and query representations.

3 Learning to Route Test-Time Economical LLM

In this section, we outline LLMRec framework, discuss the construction of the recommendation representations, implementation details, and cost constraints in deployment.

3.1 Representation of Model & Query

Motivation: The model is a black box to the router, and decisions must be made without inferring on all candidate models to minimize overhead. Directly extracting features from an LLM's high-dimensional parameters is infeasible. To address this, we construct a *model representation* that incorporates both *intrinsic properties* and *capability distributions*. This allows the router to learn how a model's potential impacts its generalization to new instructions. To achieve this, we also build a comprehensive *coreset* of diverse evaluation data. The advantage is that when a new model arrives, we can assess its core capabilities on this coreset with minimal overhead.

Model Representation For the candidate LLM f^m, we categorize its representation into intrinsic properties and capability distributions, with each dimension optional.

1. **Intrinsic properties** include model structure (*e.g.*, *publisher*, *name*, *architecture*, *number of layers*, *layer types*, *total parameters*, *training details*, *precision*, and *feature descriptions*). Additional information may include *HuggingFace download count*, *open-source licenses*, etc.
2. **Capability distributions** are divided into evaluated on offline benchmark and online coreset.
   (a) Given that most LLMs publish standard benchmark performances on release, we document model performance on benchmarks such as MMLU [24], MMLU-Pro [51], BBH [47], ARC-Challenge [6], TruthfulQA [32], Winogrande [43], and HellaSwag [60]. For reasoning capabilities, we consider domains like mathematics (MATH [25], MMLU-STEM, GSM8K [12]) and code generation (HumanEval [10], HumanEval+ [34], MBPP [4], MBPP+ [34]). For offline capabilities, we focus on the average performance across these datasets, and definitely, the router will not explicitly know which benchmark the current user query belongs to.
   (b) To expand the assessment of model capabilities, we also create an online evaluation *coreset*: it acts as a bridge linking the model's historical behavior with its future expected performance. By sampling 20-shot examples from each of the 71 categories in MMLU and MMLU-Pro, we form a core set of 1,415 instructions for online evaluation. In parallel, we select from specialized domains like mathematics, code generation, healthcare, law, and finance. We extract 5 keywords for each category as semantic descriptors and compute class-center embeddings. Detailed information is provided in the section 4.

Query Representation We divide the representation for a user instruction $\mathbf{x}_i$ into general semantic representation and meta-dimensional information.

1. **General semantic embeddings**: For a user instruction $\mathbf{x}_i$, we use 3 general encoders $\boldsymbol{\psi}$ (*e.g.*, $\text{GTE}_{\text{large}}$ [62] $\sim$ 0.33B, Qwen2.5-0.5B-Instruct [57], and RoBERTa-Large) to extract embeddings.
2. **Meta-dimensional information**, *e.g. answer difficulty*, *reasoning level*, *content diversity*, *temporal stability*, *conceptual ambiguity*, and *domain expertise*, is extracted via specific prompts by encoder Qwen2.5-7B-Instruct [57] through a few inference steps.

Context Representation We also construct contextual features to align routing decisions with service-level objectives. These include the user-specified *inference type* (*e.g.*, self-consistency, CoT, Tree-of-Thought (ToT)) and the *cost threshold*, which we discretize into five levels.

The features described above are all attributes of either the query or the model side. In RecSys, these are known as first-order features. We note that any of these feature dimensions can be optional (in implementation, they are null-padded). Modern RecSys, however, automatically constructs and learns

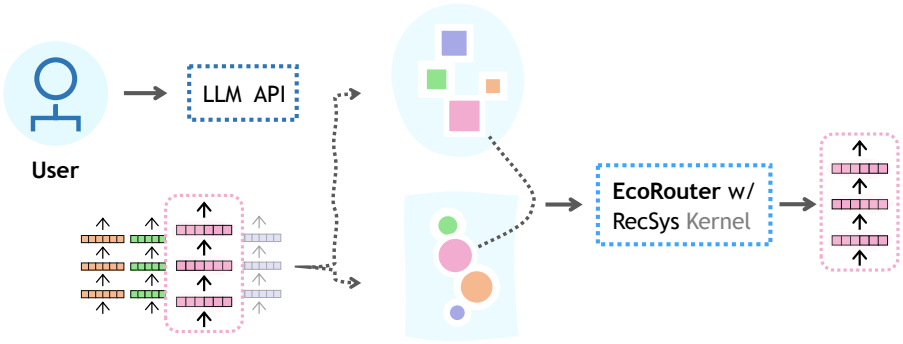


Figure 3: **The flowchart of LLMRec.** The representation of model, query, context, and behavioral sequences is constructed. Then, the feature relationships are learned by the LLMRec with the RecSys kernel to route to the corresponding LLM.

from numerous high-order cross-features to capture the complex, non-linear matching relationship between the two sides. We introduce some examples of *explicit* cross-features, such as those centered on the historical behavioral sequences of models relevant to the current query. Specifically, we consider the performance of candidate LLMs on **i**) our core set of tasks and **ii**) on the top-k nearest training neighbors of the current query. These are introduced as dynamic capability features that are more representative of and relevant to the query. While an inherent gap may exist between a model's static capabilities and a query's semantics, the LLMRec framework, like modern RecSys, is designed to learn from the complex, high-order relationships among all available model, query, contextual, and behavioral sequence features.

3.2 Routing as Recommendation

We formalize the routing problem as a ranking in RecSys. In this paradigm, the input query ($\mathbf{x}_i$) and candidate LLM (f^m) are mapped to the *user* and *item*. The user-specified context, such as the cost threshold and inference type, provides the situational context for the recommendation. The core objective is to learn a function that predicts the suitability for any given (query, model, context) tuple. This suitability is a complex, non-linear match using high-order cross-features, corresponding to the core challenge in modern RecSys. LLMRec is designed to leverage principles from established architectures to learn these interactions.

For instance, methods like Wide & Deep [11] excel by combining two components: a "wide" part that memorizes explicit, low-order feature interactions (*e.g.*, manually crafted crosses) and a "deep" part that uses MLPs to generalize and learn implicit, high-order relationships from dense embeddings. Manually designing cross-features is difficult. Models like **DeepFM** [23] automate this by integrating Factorization Machines (FM) with an MLP. The FM component efficiently models second-order feature interactions by learning a low-dimensional latent vector for every feature and taking their dot products. These latent vectors are shared with the deep component, allowing end-to-end learning. Enhancements like **AFM** [55] introduce an attention mechanism to weigh the importance of different feature interactions. Other models learn implicit interactions more directly: **FGCNN** [33] applies convolutions to the feature map to capture local interaction patterns. **FiGNN** [30] employs graph neural networks (GNNs), treating features as nodes to capture complex, high-order relationships via aggregation.

Furthermore, a model's suitability is not static; it is highly dependent on the query. We draw inspiration from behavioral modeling in RecSys, such as the Deep Interest Network (DIN) [65], which uses an attention mechanism to dynamically weigh a user's historical behaviors based on their relevance to a target item. Analogously, LLMRec treats a model's historical performance, such as on our *coreset* or on the *top-k nearest training neighbors* of the current query, as a dynamic behavioral sequence. This allows the framework to learn a query-aware representation of a model's capabilities.

By integrating first-order features (of query, model, context) and learning their complex, high-order interactions, the RecSys framework learns to accurately rank all candidate LLMs. This ranking

allows it to recommend the model that best approaches the Pareto optimum for the given query and service-level constraints.

3.3 Training Data Construction

A training instance is $(\mathbf{q}, \mathbf{m}, \mathbf{c}, \mathbf{b}) \rightarrow s$, where the representations for the query ($\mathbf{q}$), model ($\mathbf{m}$), context ($\mathbf{c}$), and behavioral features ($\mathbf{b}$) map to a target suitability score s. To learn this function, we construct a massive dataset of over 1 billion model-query pairs. This dataset spans more than 50 LLM families (from 2024) evaluated on over 30 diverse benchmarks. We capture performance logs using various inference modes, including *direct* generation, *self-consistency* [50], *Chain-of-Thought (CoT)* [52], and *Tree-of-Thought (ToT)* [58].

For each pair $(\mathbf{x}_i, \mathbf{o}_i^m)$ (instruction $\mathbf{x}_i$, model m's output $\mathbf{o}_i^m$), we record the key labels as:

1. Accuracy (Acc): Evaluated using dataset-specific metrics against the ground-truth answer $\mathbf{a}_i$.
2. Cost: Calculated as $\text{Cost}(\mathbf{o}_i^m) = c_{f_m} \times |\mathbf{o}_i^m|$, where c_{f_m} is the estimated per-token inference cost (based on model scale) and $|\mathbf{o}_i^m|$ is the length of response tokens.

To generate the ground-truth suitability ranking s, we follow a Pareto-based principle that prioritizes cost-effectiveness. For a given query, we define the preference order as:

$$\text{Ranking} = \underset{\text{Cost}(\mathbf{o}_i^m)}{\text{Sort}} \left(\{f_m\}_{\text{Acc}(\mathbf{o}_i^m, \mathbf{a}_i)>0}\right) \oplus \text{Shuffle}\left(\{f_m\}_{\text{Acc}(\mathbf{o}_i^m, \mathbf{a}_i)=0}\right) \tag{2}$$

where $\oplus$ denotes ordered concatenation. All models that correctly solve the task ($\text{Acc} > 0$) are ranked first, sorted by their $\text{Cost}(\cdot)$ in ascending order (lower cost is better). All models that fail ($\text{Acc} = 0$) are shuffled and ranked last. Crucially, this ground-truth ranking is computed relative to the input contextual features $\mathbf{c}$, especially the user-specified *cost threshold*. This teaches the LLMRec to adapt its routings, learning which LLM is "best" under different budget constraints.

4 Experiments

4.1 Implementation Details

Candidate LLMs Zoo of Training As mentioned, we consider 52 different LLMs. Among them, 32 models are under 10B parameters, 15 models are between 10B and 20B parameters, and 5 models are around 70B parameters. In practice, we test more than 80 models, but exclude early models that did not have CoT capabilities.

Domains of Training Datasets We consider general evaluation datasets, commonsense reasoning, math reasoning, code generation, symbolic reasoning, and specific domain datasets such as medical, law, and financial datasets, totaling 35 datasets.

Interaction Scaling We perform inference with each candidate LLM on the target datasets to generate nearly 10m interaction pairs. We sample approximately 1100k of these for training, with stronger-performing pairs being assigned higher sampling weights. Out of the 35 datasets, 24 are multiple-choice datasets, and 11 are fill-in-the-blank or question-answer datasets, including token generation via the generate method. For these datasets, we incorporate self-consistency, CoT, and Tree-of-Thought (ToT) reasoning modes. We also include datasets where the performance did not improve or even declined after applying complex reasoning modes.

Evaluation Metrics We evaluate the 24 multiple-choice datasets using perplexity (PPL). Most additional 11 fill-in-the-blank datasets are evaluated using regular expressions to extract the final answers. We follow the corresponding evaluation libraries for some domain-specific datasets (such as math extraction processes or code generation-type pass@k).

Candidate LLMs Zoo of Evaluation As shown in Table 1, we have mixed 5 small-scale LLMs with fewer than 10B parameters, 2 LLMs between 10B and 20B parameters, and 3 large-scale LLMs with around 70B parameters to ensure that the LLM library contains varying capabilities, from small to large.

Method	#Params	General	Comm. Reasoning			Mean
		MMLU 5-shot	TruthfulQA 0-shot	ARC-C 25-shot	MMLU-stem 5-shot	
			Small-scale LLMs (<10B)			
InternLM2.5 [8]	7.7B	69.88	54.56	60.75	65.31	62.63
Meta-Llama-3 $_{\text{Instruct}}$ [49]	8.0B	65.59	51.63	62.12	58.32	59.42
Qwen2 $_{\text{Instruct}}$ [56]	7.6B	69.13	55.49	61.43	63.45	62.38
GLM-4 [61]	9.4B	69.28	59.32	66.13	64.45	64.80
Phi-3 $_{\text{Small-128K}}$ [2]	7.4B	75.90	64.62	71.08	69.09	70.17
Best-Performing of *Small-scale LLMs*	-	75.90	64.62	71.08	69.09	70.17
			Large-scale LLMs (∼70B)			
Meta-Llama-3 $_{\text{Instruct}}$ [49]	70B	79.89	61.83	71.67	73.92	71.83
Qwen2 $_{\text{Instruct}}$ [56]	72B	83.79	54.85	68.62	**79.85**	71.78
Mixtral-8x22B $_{\text{Instruct-v0.1}}$ [26]	140B	77.63	**68.19**	72.78	71.64	72.56
Best-Performing of *Large-scale LLMs*	-	83.79	68.19	72.78	79.85	76.15
			LLM Routing			
Random Selection	∼ 32B	72.98	58.87	67.83	67.96	66.91
$\text{GTE}_{\text{Large}}$ [62]	∼ 55B	74.72	61.08	69.62	69.12	68.64
LR [42]	∼ 33B	73.25	59.61	66.04	68.12	66.76
Deep & Wide [11]	∼ 27B	82.32	67.44	71.67	77.37	74.70
Ours w/ DeepFM [23]	∼ 26B	82.29	67.81	71.93	76.74	74.69
AFM [55]	∼ **25B**	79.84	63.53	70.39	77.50	72.82
DIN [65]	∼ 31B	**83.92**	66.83	**72.95**	78.36	**75.52**

Table 1: **Comprehensive Routing Evaluation on General, Commonsense, and Reasoning Tasks.** We compare the response accuracy across various benchmarks (MMLU, TruthfulQA, ARC-C, and MMLU-stem). We categorize methods by model scale (Small-scale LLMs, Large-scale LLMs, and LLM Routing methods). We show the number of parameters ("#Params"). Bold represents the best performance, and underlined is the second-best.

Evaluation Benchmarks Similarly, we have construct evaluation dataset that includes general evaluation benchmarks like MMLU [24], TruthfulQA [32], and commonsense reasoning tasks such as ARC-Challenge [6] and MMLU-stem [24], as well as mathematical reasoning benchmarks like GSM8K [12], and symbolic reasoning tasks like BBH [47]. Different models exhibit performance variations on these datasets. We reference the scale size and FLOPs to correspond with the cost, and by multiplying the response token count, we calculate the total cost per unit on each dataset.

Model Representation Construction. As outlined in section 3, we construct model representations using the descriptions and coreset. Each dimension is segmented into multiple values, with continuous values being bucketed. These segmented values are then mapped to randomly initialized embeddings, which are incorporated into the training process.

Baseline methods. For the baseline, we consider randomly selecting from the LLM library and using all responses $\mathbf{o}i$ from the inference downstream datasets. We employ $\text{GTE}_{\text{Large}}$ to match $\mathbf{o}_i$ with the instruction $\mathbf{x}_i$, and the LLM with the highest score is selected.

Average scale calculation ("#Params" in the Tables). We average the scales of all selected LLMs based on the instruction dimensions of all evaluation data in both tables, which results in the LLM scale presented in "#Params".

Generalization to unseen models and datasets. As shown in Table 1, the models underlined in our LLM library have not appeared in the routing training set (one large-scale and one small-scale LLM). All datasets, except for MMLU, are considered unseen datasets.

Method	#Params	Math Reasoning GSM8K 4-shot, CoT		Symbolic Reasoning BBH 3-shot, CoT		Mean	
		Perf.	Leng.	Perf.	Leng.	Perf.	Leng.
Small-scale LLMs (<10B)							
InternLM2.5	7.7B	74.37	371	68.13	452	65.50	412
Meta-Llama-3 $_{\text{Instruct}}$	8.0B	56.18	273	60.93	399	59.13	336
Qwen2 $_{\text{Instruct}}$	7.6B	78.92	368	62.92	526	65.22	447
GLM-4	9.4B	79.53	505	74.43	487	68.86	496
Phi-3 $_{\text{Small-128K}}$	7.4B	82.34	449	73.94	521	72.83	485
Best-Performing	-	82.34	449	74.43	487	72.91	468
Large-scale LLMs (∼70B)							
Meta-Llama-3 $_{\text{Instruct}}$	70B	83.17	580	81.48	635	75.33	608
Qwen2 $_{\text{Instruct}}$	72B	**88.86**	535	**82.89**	593	76.48	564
Mixtral-8x22B $_{\text{Instruct-v0.1}}$	140B	84.31	553	79.54	610	75.68	582
Best-Performing	-	88.86	535	82.89	593	79.39	564
LLM Routing							
Random Selection	∼ 32B	77.71	489	73.58	534	69.82	512
$\text{GTE}_{\text{Large}}$	∼ 55B	80.52	528	74.37	580	71.57	554
LR	∼ 33B	83.40	457	73.84	532	70.71	495
Deep & Wide	∼ 27B	86.05	426	78.27	489	77.19	458
Ours w/ DeepFM	∼ 26B	87.87	411	78.02	495	77.44	453
AFM	∼ **25B**	86.05	404	79.40	476	76.12	440
DIN	∼ 31B	87.19	446	79.94	500	**78.20**	473

Table 2: **Comprehensive Routing Evaluation on Math and Symbolic Reasoning Tasks.** We compare performance on math and symbolic reasoning tasks, showing response accuracy ("Perf.") and average token usage ("Leng."). Similar to Table 1, we estimate the approximate model scale associated with each routing method. The total computational cost of a method on the data is approximately proportional to the product of the model scale ("#Params") and the average token usage ("Leng."). Bold is the best, and underlined is the second-best.

4.2 Results Analysis

Table 1 illustrates the performance of LLMRec using different recommendation system kernels on General and Commonsense Reasoning tasks. We test our routers at two scales, with small-scale LLMs showing some variation in performance. Among the small-scale LLMs, Phi-3 Small-128K achieved the best results, scoring the highest across all tests with an average of 70.17. Other small-scale LLMs, such as InternLM2.5, Meta-Llama-3 Instruct, Qwen2 Instruct, and GLM-4, also deliver decent performance.

In the large-scale LLM category, Qwen2 Instruct (72B) outperforms the others, notably earning a high score of 79.85 on MMLU-stem. Meta-Llama-3 Instruct (70B) follows closely, showing strong performance across most tests. Mixtral-8x22B Instruct-v0.1 excels in specific metrics like TruthfulQA, but its overall average score was slightly lower than the other methods. LLMRec manages the routing of instructions for all the above LLMs. Comparison methods were also used, such as random selection and embedding matching using $\text{GTE}_{\text{Large}}$ (with estimated LLM scales of 32B and 55B). Most of LLMRec's routes were in the 20B+ range. When routing LLMs with the recommendation system's DIN architecture, the best results are achieved when historical model-data interactions are included in the sequence, reaching an average accuracy of 75.52. Wide & Deep and DeepFM also achieve similarly high performance with relatively low overhead. Notably, on MMLU and ARC-Challenge, the LLM's performance surpasses the top-performing models in the datasets.

In Table 2, we present the routing capability on deep reasoning tasks, including GSM8K and BBH datasets for math and symbolic reasoning. The candidate LLM library and comparison methods are the same as in Table 1. In this evaluation, the LLMs performed inference through a generation-based

approach and assembled CoT. The "Mean" in the Table 2 represents the average across all data, including the results from Table 1. After inference, we calculate the average token length for each method. The overall cost for each method is roughly determined by multiplying the model scale by the token length. LLMRec achieves superior performance while reducing the average output length by approximately 30% compared to others, resulting in the optimal average performance.

Limitations

LLMRec has not considered market factors such as fluctuations in LLM API pricing; instead, we use the LLM parameter scale as an approximation. The generalization capability of model representations under modified pricing schemes requires further investigation. The system's evaluation of new LLMs depends on a sufficient volume of historical interaction data. This creates a cold-start challenge, a common issue for all routing systems in the rapidly evolving LLM ecosystem. This limitation may be addressed by continuously updating the coreset to enhance the validity of the model representations.

5 Conclusion

In this paper, we introduced LLMRec, a novel framework that optimizes the cost-performance trade-off for LLM API services by pioneeringly reframing model routing as a recommendation system (RecSys) task. Our core insight is that an LLM's suitability for a query, balancing capability and cost, is a complex, latent signal equal to user-item preference. LLMRec learns this signal by modeling rich, multidimensional features for models, queries, operational contexts (including user-specified cost thresholds), and behavioral sequences. Our extensive experiments demonstrate that LLMRec reduces inference costs by over 38% on average while preserving task accuracy. The framework shows robust zero-shot generalization to unseen tasks and seamless adaptation to a dynamically evolving model zoo, validating intelligent routing as a viable solution for scalable and economical LLM deployment.

Acknowledgments and Disclosure of Funding

National Key R&D Program of China (2024YFE0202800), NSFC(62376118), Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] Cost estimation of using GPT-3 for real applications. https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-\gpt-3-pricing-explained. Accessed: 2024-12-26.

[2] Marah I Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, et al. Phi-3 technical report: A highly capable language model locally on your phone. *CoRR*, abs/2404.14219, 2024.

[3] 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. In *NeurIPS*, 2024.

[4] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

[5] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *ICIP*, 2019.

[6] Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, et al. Think you have solved direct-answer question answering? try arc-da, the direct-answer AI2 reasoning challenge. *CoRR*, abs/2102.03315, 2021.

[7] Leo Breiman. Stacked regressions. *Machine learning*, 24:49–64, 1996.

[8] Zheng Cai, Maosong Cao, Haojiong Chen, et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

[9] Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *CoRR*, abs/2305.05176, 2023.

[10] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 7–10. ACM, 2016.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[13] Jiaxi Cui, Munan Ning, Zongjian Li, Bohua Chen, Yang Yan, Hao Li, Bin Ling, Yonghong Tian, and Li Yuan. Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model. *CoRR*, abs/2306.16092, 2024.

[14] Xiangxiang Dai, Jin Li, Xutong Liu, Anqi Yu, and John C. S. Lui. Cost-effective online multi-llm selection with versatile reward models. *CoRR*, abs/2405.16587, 2024.

[15] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[16] 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 *ICLR*. OpenReview.net, 2024.

[17] Tao Feng, Yanzhen Shen, and Jiaxuan You. Graphrouter: A graph-based router for LLM selections. *CoRR*, abs/2410.03834, 2024.

[18] Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-loras: An efficient multitask tuning for large language models. *CoRR*, abs/2403.03432, 2024.

[19] Roy Thomas Fielding and Richard N. Taylor. *Architectural styles and the design of network-based software architectures*. PhD thesis, 2000.

[20] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1504.

[21] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[22] M. A. Ganaie, Minghui Hu, Ashwani Kumar Malik, Muhammad Tanveer, and Ponnuthurai N. Suganthan. Ensemble deep learning: A review. *Eng. Appl. Artif. Intell.*, 115:105151, 2022.

[23] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. In Carles Sierra, editor, *IJCAI*, pages 1725–1731. ijcai.org, 2017.

[24] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[25] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*, 2021.

[26] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[27] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *CoRR*, abs/2401.04088, 2024.

[28] Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. In *NeurIPS*, 2023.

[29] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *CVPR*, 2021.

[30] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for CTR prediction. *CoRR*, abs/1910.05552, 2019. URL `http://arxiv.org/abs/1910.05552`.

[31] Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*, 2024.

[32] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *ACL*, pages 3214–3252. Association for Computational Linguistics, 2022.

[33] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. Feature generation by convolutional neural network for click-through rate prediction. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1119–1129. ACM, 2019.

[34] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

[35] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[36] Aman Madaan and Yiming Yang. FLOWGEN: fast and slow graph generation. *CoRR*, abs/2207.07656, 2022.

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

[38] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[39] Michal Pándy, Andrea Agostinelli, Jasper R. R. Uijlings, Vittorio Ferrari, and Thomas Mensink. Transferability estimation using bhattacharyya class separability. In *CVPR*, 2022.

[40] J. Ross Quinlan. Bagging, boosting, and C4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, AAAI 96, IAAI 96, Portland, Oregon, USA, August 4-8, 1996, Volume 1*, pages 725–730. AAAI Press / The MIT Press, 1996.

[41] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[42] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*, pages 521–530. ACM, 2007.

[43] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *AAAI*, pages 8732–8740, 2020.

[44] Marija Sakota, Maxime Peyrard, and Robert West. Fly-swat or cannon? cost-effective language model choice via meta-modeling. In *ACM WSDM*, pages 606–615. ACM, 2024.

[45] 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.

[46] KV Aditya Srivatsa, Kaushal Kumar Maurya, and Ekaterina Kochmar. Harnessing the power of multiple minds: Lessons learned from LLM routing. *CoRR*, abs/2405.00467, 2024.

[47] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *ACL*, pages 13003–13051. Association for Computational Linguistics, 2023.

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

[49] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[50] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net, 2023.

[51] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024.

[52] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022.

[53] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, Jinshi Huang, Charles Bai, et al. Sustainable AI: environmental implications, challenges and opportunities. In Diana Marculescu, Yuejie Chi, and Carole-Jean Wu, editors, *MLSys*. mlsys.org, 2022.

[54] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564, 2023.

[55] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *IJCAI*, pages 3119–3125. ijcai.org, 2017.

[56] An Yang, Baosong Yang, Binyuan Hui, et al. Qwen2 technical report, 2024. URL `https://arxiv.org/abs/2407.10671`.

[57] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024.

[58] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*, 2023.

[59] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021.

[60] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *ACL*, pages 4791–4800, 2019.

[61] Aohan Zeng, Bin Xu, Bowen Wang, et al. Chatglm: A family of large language models from GLM-130B to GLM-4 all tools. *CoRR*, abs/2406.12793, 2024.

[62] Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, et al. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval. *arXiv preprint arXiv:2407.19669*, 2024.

[63] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. Model spider: Learning to rank pre-trained models efficiently. In *NeurIPS*, 2023.

[64] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Class-incremental learning: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(12):9851–9873, 2024.

[65] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *ACM SIGKDD*, pages 1059–1068. ACM, 2018.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly summarize the paper's core contributions and define the scope of the research presented, accurately reflecting the content discussed in the main body.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discussed the limitations in the conclusion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper is empirical and does not contain theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In the paper, we have provided a detailed description of the model structure and training method, and in the experimental section, we have detailed the evaluation protocol and the datasets used for comparison.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code for the methods is provided in the supplemental material.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We specified the training pipeline, the evaluation settings and protocol, and the evaluation datasets in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the statistical significance of the experiments in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In the appendix, we describe the computational resources used to complete model training and experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The paper is compliant with the NeurIPS Code of Ethics in all aspects.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This work has no potential negative societal impacts. The topic of the paper is using large models for table understanding and question answering, which can be applied in areas such as finance, healthcare, and science to improve productivity.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer:[NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.