IR3DE: A Linear Router for Large Language Models

Eros Fanì [1] **Oğuzhan Ersoy** [1]

Abstract

Foundational Large Language Models (LLMs) demonstrate proficiency on a wide range of general tasks, and achieve remarkable results on various specialized tasks via domain-expert LLMs. With the ever-growing list of available LLMs, inference routers are being proposed to select the most appropriate LLM for each prompt. However, existing routing methods either optimize cost across weak-to-strong generalist LLMs or require substantial training to support domain-expertise routing. In this paper, we propose IR3DE, a Ridge Regression-based Router for Domain Experts that provides cheap and fast routing decisions for each prompt. We evaluate IR3DE in two Causal Language Modeling (CLM) settings where the tasks are next-token prediction for all domains, and one reasoning setting where each domain has its own distinct reasoning task. Despite being a linear router, IR3DE achieves performance comparable to the other baselines in both CLM settings, and surpassing them in the reasoning setting, with a normalized performance of 98.4%. Moreover, IR3DE enables the addition or removal of new domain experts without requiring the router to be retrained from scratch, allowing a dynamic set of LLMs to be served with minimal disruption to the router itself. Our code is available here: `https://github.com/gensyn-ai/IR3DE`.

1. Introduction

Large Language Models (LLMs) have shown remarkable performance gains on both language and cognitive tasks in recent years. Foundation (or generalized) models perform well across a wide range of tasks (Radford et al., 2019; Touvron et al., 2023; Bai et al., 2023), whereas expert (or specialized) ones excel on specific tasks such as code generation (Chen, 2021), mathematical problem-solving (Cobbe et al., 2021), and instruction following (Zhou et al., 2023). Depending on a user's needs or the prompt, different models can be used. Routing all queries to a generalist model can be suboptimal because such models may lack the necessary domain knowledge or be unnecessarily expensive. Thus, it is beneficial to route each query to the most relevant (expert) model.

LLM routing allows us to dynamically assign each query to the most appropriate model based on selection criteria. A common selection criterion is the cost-performance trade-off between weak and strong LLMs, where the models have similar capabilities but differ in capacity (Ong et al., 2025; Feng et al., 2025; Song et al., 2025; Jitkrittum et al., 2025). Here, the router makes the selection mainly based on the difficulty of the query, i.e., sending difficult queries to the strong (but large) model and vice versa, rather than on their expertise. There are also routers that assign queries based on each LLM's expertise, aiming to maximize accuracy (Simonds et al., 2024; Stripelis et al., 2024). However, such expert routers utilize additional (language) models either to classify each query or to obtain token embeddings using the last hidden layer representations. Therefore, they require collecting domain datasets to train the router, which may not be feasible due to privacy concerns, or they use a language model as the router.

In this paper, we propose IR3DE, which enables efficient routing to the most suitable domain expert. IR3DE utilizes a closed-form solution via ridge regression, inspired by its use in federated learning (Afonin & Karimireddy, 2021; Cai et al., 2022; Huang et al., 2022; Fanì et al., 2024) and MoE routers (Fanì & Ersoy, 2026). We use a linear ridge-regression token router and select the expert LLM based on the confidence of the top-k tokens. Since the domain statistics can be computed asynchronously for the solution, IR3DE does not require collecting the datasets in a single location. Moreover, the cost of running IR3DE is irrelevant, as it only requires token embedding with any desired embedding layer and the inversion of a small matrix (typically about a 1k-by-1k matrix) to be performed only one time.

Our contributions can be summarized as follows:

- We introduce IR3DE, a linear expert router for LLMs. IR3DE router consists of two components: a *token router* (TR) based on ridge regression and a *sample route selector* (SRS). Thanks to its linear construction, IR3DE is cheaper and faster than LM-based baselines. Moreover, it enables the addition or removal of new

[1]Gensyn. Correspondence to: Eros Fanì <eros@gensyn.ai>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

domain experts without requiring the router to be retrained from scratch.

- We evaluate IR3DE in two CLM settings and one reasoning setting, across several domains and tasks. IR3DE achieves performance comparable to the other baselines in both CLM settings, and surpasses them in the reasoning setting.
- We present three variants of IR3DE (specifically, of the SRS) based on averaging, majority voting, or entropy of TR outputs. Experimental results show that the entropy-based approach achieves the best performance on complex reasoning tasks, where precision becomes more crucial.

2. Related Work

With the increasing number of LLMs, including both generalist and expert models, routing has become increasingly important for dynamically assigning each query to the most appropriate model. Early work on LLM routing primarily focused on the cost–quality trade-off in fixed pools of generalist models (Ding et al., 2024; Ong et al., 2025; Song et al., 2025). By estimating prompt difficulty, these methods route queries to an adequate model rather than always invoking the most powerful (and costly) one. In addition to a single LLM selection, cascading approaches have been proposed in (Chen et al., 2024; Zheng et al., 2025; Dekoninck et al., 2025), where first a weaker LLM generates the output and depending on the quality or confidence, it is escalated to a stronger LLM. Subsequent work has broadened this paradigm by taking into account the reasoning strategies and task profiles (Shao et al., 2026; Xue et al., 2026; Liu et al., 2026), and by enabling routers to accommodate previously unseen models without retraining from scratch (Feng et al., 2025; Jitkrittum et al., 2025; Wang et al., 2026).

Another selection criterion for routing decisions is accuracy. In accuracy-oriented routing, where the router assesses the domain of a prompt and forwards it to the most relevant (specialist) model (Simonds et al., 2024; Stripelis et al., 2024). Finally, there are extensions using multiple experts: Symbolic-MoE (Chen et al., 2025) selects several experts for a given query and combines their outputs using an aggregator, and HierRouter (Gupta et al., 2025) proposes a hierarchical routing approach that selects expert models in a multi-hop inference setting.

The most relevant works are *MoDEM* (Simonds et al., 2024) and *PolyRouter* (Stripelis et al., 2024). The MoDEM router uses a DeBERTa v3 model (He et al., 2021) trained on the union of all domain datasets used to train the domain experts. Because of this, this method is unsuitable in cases where it is not possible to collect data from all domains on a single node due to privacy constraints or limited communication budget, or when the computational or memory budget is insufficient for additional training. PolyRouter presents multiple routing methods: (i) *BERT-router* that trains a BERT model (Devlin et al., 2019) to classify the domains; (ii) *MLP-router* that trains a 2-layer perceptron using a Bag-of-Words representations of training queries; and (iii) *1NN-router*, which uses sentence-transformer embeddings to find the nearest training query and select the corresponding expert. All these methods require additional (language) models either to classify each query or to obtain token embeddings.

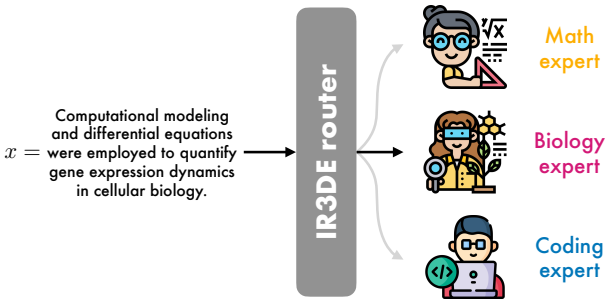


Figure 1. Given an input text x, our IR3DE router selects the most appropriate expert solely based on the token embeddings.

3. Method

We have access to a set of LM experts $\{f_d\}_{d=1}^{D}$. Each expert has been trained on a different domain dataset $\mathcal{D}_d$, $d = 1 \dots D$ for some upstream or downstream task (*e.g.,* math, biology, coding, ...), whose samples belong to different distributions. Our objective is to construct a routing mechanism that, given an input text, is able to select the expert who would maximize the performance for that input. Performance could be any desired metric, depending on the tasks for which the experts have been trained. For instance, for causal language modeling, the desired metric could be perplexity, while for coding it could be pass@1. The setting is represented in Figure 1.

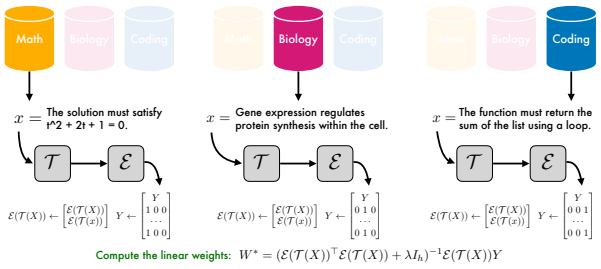
Our IR3DE router is constituted of two components: a Token Router and a Sample Route Selector. We provide an overview of our IR3DE method in Figure 2.

3.1. Token Router (TR)

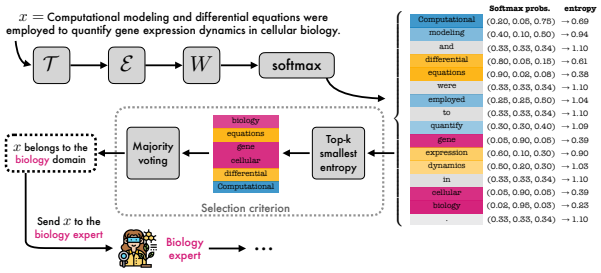
Given an input x, we first construct the Token Router $\mathcal{R}$:

$$\mathcal{R}(x) = softmax(\mathcal{E}(\mathcal{T}(x))W).$$

Without loss of generality, assume that the input text x is composed of T tokens. The tokenizer $\mathcal{T}$ returns a list of token ids, which are then forwarded to a pre-trained embedding layer $\mathcal{E}$ to construct a matrix of embeddings $\mathcal{E}(\mathcal{T}(x)) \in \mathbb{R}^{T \times h}$. Finally, linear weights $W \in \mathbb{R}^{h \times C}$ are multiplied to the matrix of embeddings, to construct softmax probabilities for each token. Therefore, each input token will be associated to a softmax probability vector $\mathcal{R}(x)_t$. Please note that any choice for the tokenizer $\mathcal{T}$



(a) Token Router.



(b) Sample Route Selector.

Figure 2. **(a)** Construction of the Token Router. Input samples are forwarded to the tokenizer and the embedding layer. The embeddings and one-hot encoding vectors are stacked together and eventually used to compute the optimal RLS weights. **(b)** At inference time, the input samples are forwarded to the token router to extract softmax probabilities associated with each token. Then, entropy is computed for the softmax probabilities of each token, and only the top-k tokens with the smallest entropy are retained (In the image, k=6). Finally, each of the surviving tokens cast a vote to elect the final expert to whom x will be forwarded.

and embedding layer $\mathcal{E}$ works, given that the $\mathcal{E}$ has been trained using the tokenizer $\mathcal{T}$. In other words, their choice is independent from the tokenizers and embedding layers actually used by the experts, which could also vary.

Now, let $\mathcal{E}(\mathcal{T}(X)) \in \mathbb{R}^{n \times C}$ be the matrix of stacked embeddings of all the tokens of each sample of each domain dataset, and $Y \in \mathbb{R}^{n \times C}$ be the matrix of stacked one-hot vectors associated with each sample, such that $Y_{ij} = 1$ iff the original sample of token i belongs to domain j, 0 otherwise. We obtain the linear weights W by solving the regularized least squares problem $\min_{W \in R^{h \times C}} \left[\|\mathcal{E}(\mathcal{T}(X)) - Y\|^2 + \lambda \|W\|^2 \right]$, which admits the following closed form solution:

$$W^* = (\mathcal{E}(\mathcal{T}(X))^\top \mathcal{E}(\mathcal{T}(X)) + \lambda I_h)^{-1} \mathcal{E}(\mathcal{T}(X))Y,$$

where $\lambda \in \mathbb{R}^+$ controls Tikhonov regularization, and I_h is the identity matrix of size $h \times h$. To avoid large matrix multiplications, it is possible to collect RLS statistics batch-wise by exploiting the properties of the closed-form RLS solution. For instance, for batches j of J tokens, we can equivalently compute:

$$A := \sum_j [\mathcal{E}(\mathcal{T}(X))]_{j:j+J}^\top [\mathcal{E}(\mathcal{T}(X))]_{j:j+J},$$
$$B := \sum_j [\mathcal{E}(\mathcal{T}(X))]_{j:j+J}^\top [Y]_{j:j+J},$$
$$W^* = (A + \lambda I_h)^{-1} B.$$

This formulation also demonstrates that our solution is suitable for decentralized settings where domain datasets cannot be easily collected on a single node, since each dataset can be treated as a separate batch (or multiple separate batches). Moreover, it allows new expert models to join at any time without having to reconstruct the router from scratch. Alternatively, it is also possible to train $\mathcal{R}$ with any desired loss function, *e.g.,* using the cross-entropy loss, and using $\mathcal{E}(\mathcal{T}(X))$ and Y as the dataset.

3.2. Sample Route Selector (SRS)

Given an input text x, the Token Router returns a matrix of T softmax probabilities $s = [s_1^\top, s_2^\top, \ldots, s_T^\top]^\top$, where $s_t = \mathcal{R}(x)_t$ for each token t of x. Then, the Sample Route Selector computes the entropy vector $e \in \mathbb{R}^T$, such that $e_t = -\sum_{d=1}^{D} s_{td} \log s_{td}$ is the Shannon entropy of s_t. SRS then selects the tokens with the $\min(k, T)$ smallest entropy in e, returning $\hat{s} \in \mathbb{R}^{\min(k,T)}$. Finally, SRS computes the $\arg\max$ of each row of $\hat{s}$, returning the *vote* cast by each token of x of where to route the input text x, and the final expert to which x is routed is selected via majority voting.

Motivations on the Sample Route Selector construction are provided in Section 4.2.

Variants. In addition to our chosen SRS, we also present two additional variants:

- IR3DE-*all*: in this variant, instead of top-k, we use all tokens in each prompt (and the number of tokens per prompt is capped at 1024 in our experiments). Therefore, in this scenario, there is no entropy-based filtering, and all tokens contribute to selecting the final expert via majority voting. In Section 4.2, we explain how this can introduce noise into the majority-voting process.
- IR3DE-*avg*: in this variant, we modify our SRS by first computing the average of the token embeddings and selecting the final domain with the argmax of the softmax probabilities computed on the average token embedding vector. This is a cheap variant that does not require the SRS but could hinder the single tokens' ability to determine the correct routing domain, since a strong signal compression is applied.

4. Experiments

4.1. Setup

We conduct experiments in three settings: two *Causal Language Modeling* settings, CLM and CLMLARGE, where the task is next-token prediction for all domains, and one

Table 1. Datasets used for evaluating the reasoning experiments, and their evaluation metrics.

Dataset	Description	Metric
HumanEval (Chen, 2021)	Code generation benchmark consisting of programming tasks specified by natural-language prompts and associated unit tests.	pass@1: fraction of tasks solved correctly by the first generated completion.
GSM8k (Cobbe et al., 2021)	Grade-school mathematics benchmark containing linguistically diverse word problems that require multi-step reasoning.	Accuracy: fraction of examples for which the final answer is correct.
M_ARC (Lai et al., 2023)	Multilingual version of the ARC challenge benchmark, consisting of multiple-choice question-answering tasks that test reasoning ability.	Accuracy: percentage of questions answered correctly.
IFEval (Zhou et al., 2023)	Instruction-following benchmark for LLMs, with prompts containing verifiable constraints that allow automatic evaluation of compliance.	Task compliance rate: percentage of responses that satisfy all instructions.

Reasoning setting (REASONING), where each domain has its own distinct reasoning task. Below, we present the details of our settings and the baselines compared with our method. All experiments run on an NVIDIA H100 GPU with 80 GB of HBM3 memory.

Expert Models and Domain Datasets. In the CLM setting, we train our expert models starting from a shared base model. Both the base and expert models are trained using the hyperparameters and datasets outlined in (Ersoy et al., 2025). Initially, we pre-trained a 115M Llama3 seed model (Grattafiori et al., 2024) on the OpenWebText corpus dataset (Gokaslan & Cohen, 2019), then finetuned the expert models on M2D2 domains (Reid et al., 2022) (coding, mathematics, physics, history and events, and philosophy and thinking). For CLMLARGE experiments, we finetuned expert models using a 1B Llama3 base model on these domains: mathematics (OpenWebMath (Paster et al., 2023)), biology (peS2o (Soldaini & Lo, 2023)), legal (Pile of Law (Henderson et al., 2022)), and dialogue (UltraChat 200k (Ding et al., 2023)). In both CLM settings, the metric used for evaluation is perplexity.

For the REASONING experiments, we use domain-specific LLama3-3B experts from MergeBench (He et al., 2025) on the following domains: coding, mathematics, multilingual understanding, and instruction following. We use the following downstream task datasets for the evaluation: HumanEval (Chen, 2021) for coding, GSM8k (Cobbe et al., 2021) for mathematics, M_ARC (Lai et al., 2023) for multilingual understanding, and IFEval (Zhou et al., 2023) for instruction following. Details on the evaluation metrics for each REASONING dataset can be found in Table 1.

Following the same practice as in (Fanì & Ersoy, 2026), we report the final results using normalized metrics relative to the performance each domain expert achieves in their respective domain (across all settings, the higher, the better).

Baselines. We compare our method with the following baselines:

- *"Domain" expert*: as in (Hu et al., 2024; Stripelis et al., 2024), we compare our proposed method against domain experts' performance across all domains.
- *Experts average*: similarly, we compare our method against the average of the expert models.
- *Random routing*: also used by (Stripelis et al., 2024) as a lower bound, the expert is randomly selected for any given prompt.
- *MoDEM*: proposed by (Simonds et al., 2024), it uses a DeBERTa v3 model (He et al., 2021) trained on the domain datasets. We present two versions of this baseline: *MoDEM-small*, which uses a DeBERTa v3 small model with 44M parameters, and *MoDEM-large*, which uses a DeBERTa v3 large model with 304M parameters.
- *1NN router*: proposed by (Stripelis et al., 2024), it uses the BERT model (Devlin et al., 2019) to extract embeddings for each input prompt. Then, at inference time, the test samples are embedded using the same BERT model, and each test sample's embedding is compared with all training embeddings using cosine similarity. Finally, the nearest training embedding is selected, and the test sample is forwarded to the expert with the corresponding domain.
- *kNN router*: expands the *1NN router* strategy by selecting k experts, and then letting them elect the winning domain via majority voting. Like the *1NN router*, it uses a BERT model to extract the embeddings.

Note that we omit the two other baselines proposed in (Stripelis et al., 2024), namely *BERT-router* and *MLP-router*, given that *MoDEM* provides a similar solution using a language model, DeBERTa v3, that is supposed to perform better than a BERT model. Additionally, please observe that *MoDEM-large* router is even larger than the experts in the CLM setting, making this baseline too expensive and impractical for real-world deployment.

For the *kNN router*, in all our settings, we tried all k $\in 1, 5, 10$. Moreover, for our IR3DE, we tried all $k \in 1, 2, 5, 10, 20, 50, 100, 200, 500$ for our top-k entropy-based

Table 2. Results in the CLM setting.

Method	Coding	Math	Physics	History	Philosophy	Average
Coding expert	100.0	76.1	84.9	99.4	85.8	86.3
Math expert	93.5	100.0	84.7	79.7	81.1	87.8
Physics expert	86.9	68.2	100.0	75.1	86.0	85.3
History expert	74.2	49.3	74.6	100.0	98.7	79.4
Philosophy expert	75.6	51.4	75.5	98.1	100.0	80.1
Experts average	90.9	72.2	88.4	93.5	94.4	87.9
Random router	84.9	67.0	84.1	89.1	90.3	83.1
MoDEM-small	97.0	95.9	94.2	98.5	**102.0**	97.6
MoDEM-large	97.4	95.5	97.3	99.4	**102.0**	98.3
kNN router	**99.9**	100.2	99.5	**99.5**	100.7	**100.0**
IR3DE-avg (ours)	**99.9**	94.4	100.6	97.5	97.5	98.2
IR3DE-all (ours)	99.8	**101.6**	**101.2**	98.5	98.6	**100.0**
IR3DE (ours)	99.1	94.6	99.5	98.1	98.6	98.2

Table 3. Results in the CLMLARGE setting.

Method	Math	Biology	Legal	Dialogue	Average
Math expert	100.0	91.5	41.2	89.5	74.1
Biology expert	71.4	100.0	30.9	65.4	65.4
Legal expert	71.4	86.0	100.0	89.5	91.8
Dialogue expert	66.4	87.8	33.0	100.0	73.6
Experts average	79.8	93.5	49.7	89.5	77.5
Random router	75.7	88.3	45.1	87.2	73.5
MoDEM-small	93.5	89.0	86.2	84.2	86.5
MoDEM-large	97.5	95.8	85.8	79.4	87.0
kNN router	98.4	**96.2**	**98.8**	98.8	**97.9**
IR3DE-avg (ours)	88.1	90.3	82.7	**99.4**	90.8
IR3DE-all (ours)	86.6	90.1	87.0	98.8	92.0
IR3DE (ours)	**98.5**	95.8	92.5	97.7	95.3

Table 4. Results in the REASONING setting.

Method	Math	Multilingual	Coding	Instruction	Average
Math expert	100.0	95.7	78.4	43.8	79.5
Multilingual expert	6.1	100.0	73.1	21.2	50.1
Coding expert	11.7	99.1	100.0	55.2	66.5
Instr. following expert	28.9	99.7	80.1	100.0	77.2
Experts average	45.9	100.0	83.9	53.4	70.8
Random router	37.7	98.4	83.8	59.3	69.8
MoDEM-small	29.5	98.6	69.5	100.3	74.5
MoDEM-large	29.5	96.7	67.7	95.3	72.3
kNN router	95.8	99.5	**96.3**	98.7	97.6
IR3DE-avg (ours)	97.3	**99.9**	85.6	**101.2**	96.0
IR3DE-all (ours)	91.5	99.1	89.2	100.1	95.0
IR3DE (ours)	**98.4**	**99.9**	94.5	100.6	**98.4**

selection rule.[1] For both *kNN router* and IR3DE, in the following tables, we show only the best-performing result.

Metrics. Following the same practice as in (Fanì & Ersoy, 2026), we report the final results using normalized metrics relative to the performance each domain expert achieves in their respective domain (across all settings, the higher, the better). In particular, for both the causal language modeling settings, for each method, and for each domain score, we divide the perplexity score of the expert associated with that domain ($\hat{p}_d$) by the perplexity score achieved by the method in that domain (p_d): $\bar{p}_d = \frac{\hat{p}_d}{p_d}$. For REASONING, we adopt a similar approach, but in this case we invert the equation, given that in the REASONING setting a higher (non-normalized) final score means a better performance: $\bar{p}_d = \frac{p_d}{\hat{p}_d}$. Finally, all our normalized scores are presented as percentages (so we multiply all the values by 100). With these definitions, scores above 100 are possible because of randomness in the generation process, as detailed in the following section.

In the following section, we present the experimental results. In all experiments, the best and second best results (excluding the corresponding expert of each domain) are highlighted in **bold** and underlined, respectively.

4.2. Results

Causal Language Modeling results. Tables 2 and 3 present the results for the CLM and CLMLARGE settings, respectively. For *kNN router*, we report results for k $= 10$ in the CLM setting, and k $= 1$ in the CLMLARGE setting; for IR3DE, we report results with $k = 100$ in the CLM setting, and $k = 10$ for the CLMLARGE setting. These values provided the best results for these methods.

As shown in the tables, our proposed methods are competitive with the baselines and, unlike *kNN-router* and *MoDEM*, do not require either an additional language model for embedding generation or the centralization of domain datasets. In particular, in CLM, they even surpass all of them in the Coding, Math, and Physics domains, and also in terms of average performance. It is worth noting that, in this setting, both *kNN router* and our IR3DE-all achieve an average performance of 100.0, meaning that, on average, they perform as well as each expert domain does in its own domain.

REASONING. Table 4 shows the results in the REASONING setting. For *kNN router*, we report results for k $= 1$; for IR3DE, we report results with $k = 10$, as these values provided the best results for these methods.

In this setting, IR3DE achieves the best average performance and the best or second-best performance across all single domains. In particular, it achieves a normalized average performance of 98.4. Comparably, *kNN router*, the second-best performing method in terms of average performance, achieves 97.6. Notably, some of the scores exceed 100, indicating performance better than that of the domain expert in their respective domains. In principle, this should not be possible, since regardless of the router's decisions, only one of the original domain experts handles each sample. However, given that the text has a random component (we use a temperature of 0.7 for all text generation), this is actually possible by chance when the routing accuracy, *i.e.,*, the percentage of samples correctly routed to their corresponding expert, approaches 100.

In the next section, we analyze the routing accuracy of IR3DE for various values of k.

[1] We use "k" for *kNN router* and "k" for IR3DE, to emphasize that the two are used in different contexts and with different meanings.

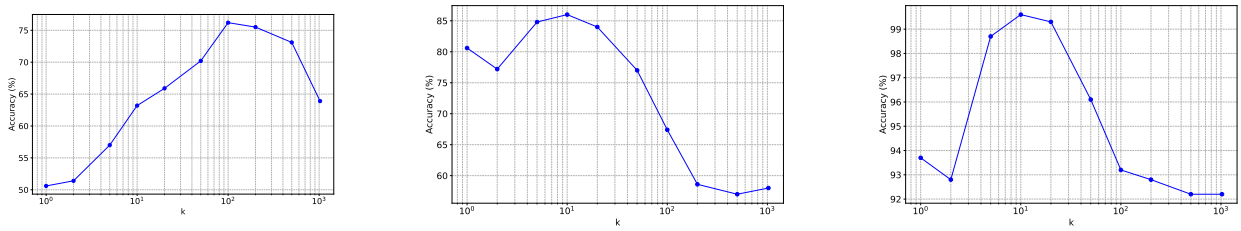


Figure 3. IR3DE routing accuracy (%) with various values of k for the top-k smallest entropy token filtering, on the CLM (left), CLMLARGE (center) and REASONING (right) settings.

On the Sample Route Selector routing mechanism of IR3DE. The Sample Route Selector in IR3DE filters tokens based on entropy. In particular, only the k tokens, whose softmax predictions are among the k smallest, are used to cast a final vote to determine the routing expert. This section answers the following question: *Why not allow all the tokens to participate in majority voting?* In the previous sections, we have answered this question empirically, by showing that IR3DE-all performs worse than IR3DE in all three settings in terms of the average score. The motivation is that, when constructing the Token Router with ridge regression, the same (common) tokens can have different domain labels, as they likely appear in different domains. For instance, it is very likely that the token "the" will appear in all domains. However, when solving the RLS problem, having the same input vector repeated with different labels will yield uncertain predictions. For example, if there are only two domains, and the token "the" appears the same number of times in each domain, the probability score of the Token Router for "the" would be the vector $(0.5, 0.5)$. Therefore, the motivation for filtering based on entropy is that the tokens for which the Token Router is more uncertain are likely the ones that appear with equivalent frequency across all domains. Such tokens are therefore not discriminative and should be excluded from the decision-making process, as they will have a higher entropy score.

Figure 3 confirms our intuition. In this figure, we present the routing accuracy (i.e., the percentage of prompts correctly routed to their corresponding domains) in all three settings with various values of k. As can be observed, the three settings follow the same trend. When only a bunch of tokens participate in the majority voting, the signal from each prompt is too shallow and IR3DE provides a smaller routing accuracy. Similarly, when too many tokens participate, they introduce noise into the selection of the routing domain, since tokens with uncertain predictions are allowed to vote alongside those with more confidence. The best solution is always to allow a sufficiently large pool of confident tokens to participate, while excluding the uncertain ones.

5. Conclusion

As the number and diversity of available LLMs continue to grow, effective routing becomes increasingly important for selecting the most suitable model for each query. Existing routing methods have largely focused on cost–performance trade-offs among generalist models. Recently proposed routers also consider domain accuracy, assigning queries to specialist models based on their domain relevance. However, existing work often relies on language-model-based classifiers or embedding models, which increase routing cost and typically require access to domain datasets for training, raising privacy concerns. These limitations underscore the need for lightweight, adaptable routing methods for expert LLM selection. Despite being a linear router, IR3DE achieves performance comparable to other baselines across all settings and even surpasses the second-best baseline, *k-NN router*, in the REASONING setting, achieving a normalized performance of 98.4%. Moreover, in CLM, IR3DE-all performs, on average, on par with expert domain models evaluated on their own domains. Together, these results highlight the strengths of IR3DE and its variants as a cheap and fast alternative for inference routing.

While IR3DE is lightweight and efficient thanks to its linear construction, it is also less expressive than stronger LLM-based routers; as a result, it may be less effective on queries that require richer semantic understanding or complex decision boundaries. This trade-off motivates several directions for future work. First, the current ridge-regression formulation could be extended to kernel ridge regression to capture non-linear structure while retaining much of the method's analytical simplicity. Second, it would be valuable to evaluate and adapt the router for more complex reasoning tasks, where domain relevance alone may be insufficient, and routing may need to account for multi-step reasoning requirements. Third, future versions of the router could explicitly incorporate system-level costs into the routing objective, including not only predictive performance but also computation, latency, and memory usage, enabling more practical deployment in resource-constrained settings.

References

Afonin, A. and Karimireddy, S. P. Towards model agnostic federated learning using knowledge distillation. *arXiv preprint arXiv:2110.15210*, 2021.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report. *CoRR*, abs/2309.16609, 2023.

Cai, J., Liu, X., Yu, Z., Guo, K., and Li, J. Efficient vertical federated learning method for ridge regression of large-scale samples. *IEEE Transactions on Emerging Topics in Computing*, 11(2):511–526, 2022.

Chen, J. C., Yun, S., Stengel-Eskin, E., Chen, T., and Bansal, M. Symbolic mixture-of-experts: Adaptive skill-based routing for heterogeneous reasoning. *CoRR*, abs/2503.05641, 2025.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024, 2024.

Chen, M. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dekoninck, J., Baader, M., and Vechev, M. T. A unified approach to routing and cascading for llms. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., and Awadallah, A. H. Hybrid LLM: cost-efficient and quality-aware query routing. In *ICLR*. OpenReview.net, 2024.

Ding, N., Chen, Y., Xu, B., Qin, Y., Zheng, Z., Hu, S., Liu, Z., Sun, M., and Zhou, B. Enhancing chat language models by scaling high-quality instructional conversations, 2023.

Ersoy, O., Kolehmainen, J., and Andrade, G. P. HDEE: Heterogeneous domain expert ensemble. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=5ukL6nPcYe.

Fanì, E. and Ersoy, O. Training-free dynamic upcycling of expert language models. *CoRR*, abs/2603.29765, 2026.

Fanì, E., Camoriano, R., Caputo, B., and Ciccone, M. Accelerating heterogeneous federated learning with closed-form classifiers. *Forty-first International Conference on Machine Learning (ICML)*, 2024.

Feng, T., Shen, Y., and You, J. Graphrouter: A graph-based router for LLM selections. In *ICLR*. OpenReview.net, 2025.

Gokaslan, A. and Cohen, V. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gupta, N., Guo, B., Kannan, R., and Prasanna, V. K. Hierrouter: Coordinated routing of specialized large language models via reinforcement learning. *CoRR*, abs/2511.09873, 2025.

He, P., Gao, J., and Chen, W. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

He, Y., Zeng, S., Hu, Y., Yang, R., Zhang, T., and Zhao, H. Mergebench: A benchmark for merging domain-specialized LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=rw50iUoyLu.

Henderson, P., Krass, M. S., Zheng, L., Guha, N., Manning, C. D., Jurafsky, D., and Ho, D. E. Pile of law: Learning responsible data filtering from the law and a 256gb open-source legal dataset, 2022. URL https://arxiv.org/abs/2207.00220.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. Routerbench: A benchmark for multi-llm routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Huang, L., Li, Z., Sun, J., and Zhao, H. Coresets for vertical federated learning: Regularized linear regression and k-means clustering. *Advances in Neural Information Processing Systems*, 35:29566–29581, 2022.

Jitkrittum, W., Narasimhan, H., Rawat, A. S., Juneja, J., Wang, Z., Lee, C., Shenoy, P., Panigrahy, R., Menon, A. K., and Kumar, S. Universal model routing for efficient LLM inference. *CoRR*, abs/2502.08773, 2025.

Lai, V., Nguyen, C., Ngo, N., Nguyen, T., Dernoncourt, F., Rossi, R., and Nguyen, T. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 318–327, 2023.

Liu, H., Zou, B., Chen, K., Liu, J., Wang, W., and Li, H. Task-aware llm routing with multi-level task-profile-guided data synthesis for cold-start scenarios. *arXiv preprint arXiv:2604.09377*, 2026.

Ong, I., Almahairi, A., Wu, V., Chiang, W., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. Routellm: Learning to route llms from preference data. In *ICLR*. OpenReview.net, 2025.

Paster, K., Santos, M. D., Azerbayev, Z., and Ba, J. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Reid, M., Zhong, V., Gururangan, S., and Zettlemoyer, L. M2d2: A massively multi-domain language modeling dataset. *arXiv preprint arXiv:2210.07370*, 2022.

Shao, C., Liu, X., Lin, Y., Xu, F., and Li, Y. Route-and-reason: Energy-efficient scaling of LLM reasoning via reinforced model routing. In *WWW*, pp. 9551–9562. ACM, 2026.

Simonds, T., Kurniawan, K., and Lau, J. H. Modem: Mixture of domain expert models. In *ALTA*, pp. 75–88. Association for Computational Linguistics, 2024.

Soldaini, L. and Lo, K. peS2o (Pretraining Efficiently on S2ORC) Dataset. Technical report, Allen Institute for AI, 2023. ODC-By, https://github.com/allenai/pes2o.

Song, W., Huang, Z., Cheng, C., Gao, W., Xu, B., Zhao, G., Wang, F., and Wu, R. Irt-router: Effective and interpretable multi-llm routing via item response theory. *CoRR*, abs/2506.01048, 2025.

Stripelis, D., Hu, Z., Zhang, J., Xu, Z., Shah, A. D., Jin, H., Yao, Y., Avestimehr, S., and He, C. Polyrouter: A multi-llm querying system. *CoRR*, abs/2408.12320, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Wang, C., Li, H., Zhang, Y., Chen, L., Chen, J., Jian, P., Zhang, Q., and Hu, S. Icl-router: In-context learned model representations for LLM routing. In *AAAI*, pp. 33413–33421. AAAI Press, 2026.

Xue, J., Lou, Q., Xing, J., and Huang, H. R2-router: A new paradigm for LLM routing with reasoning. *CoRR*, abs/2602.02823, 2026.

Zheng, H., Xu, H., Lin, Y., Fan, S., Chen, L., and Yu, K. Disrouter: Distributed self-routing for LLM selections. *CoRR*, abs/2510.19208, 2025.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

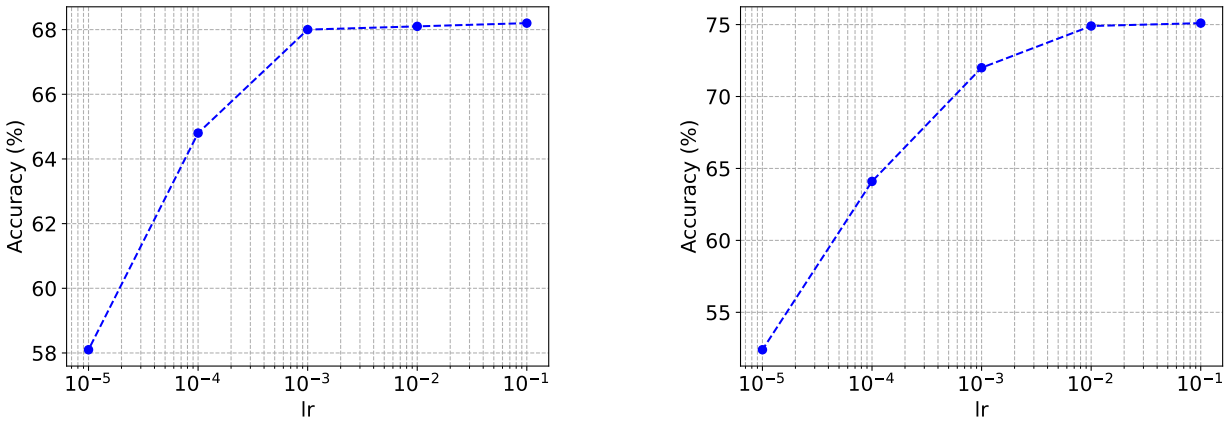


Figure 4. Final accuracy (%) of the MoDEM router with various values for the learning rate (*lr*) in the CLM setting: small router (left) and large router (right).

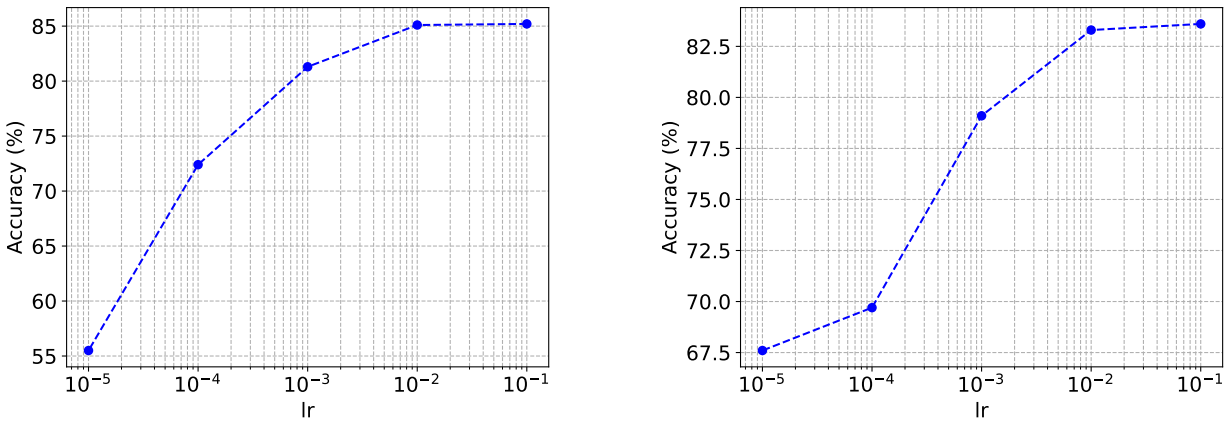


Figure 5. Final accuracy (%) of the MoDEM router with various values for the learning rate (*lr*) in the REASONING setting: small router (left) and large router (right).

A. Details on MoDEM router training

We have trained the MoDEM routers for both CLM and REASONING using a merged dataset across all domains, in the corresponding settings. For each domain, we fixed the number of samples at 1750 to avoid class imbalance. In the REASONING setting, we trained both the small and large MoDEM routers for 100 epochs. Similarly, we trained the small MoDEM in the CLM setting for 100 epochs, whereas we found it sufficient to train the large MoDEM for only 10 epochs, as the best validation routing accuracy plateaued after a few epochs. For both the large and small MoDEM, in both settings, we trained MoDEM with a batch size of 16, clipped the gradient norms to 1, and a cosine annealing scheduler with a warmup. Figures 4 and 5 show the final test routing accuracies in both the CLM and REASONING settings, respectively, with $lr \in 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$. In all cases, we eventually selected the MoDEM router trained with the largest learning rate, *i.e.,* $lr = 0.1$, as it provided the best test routing accuracy.