LARGE LANGUAGE MODEL ROUTING WITH BENCHMARK DATASETS

Anonymous authors
Paper under double-blind review

ABSTRACT

There is a rapidly growing number of open-source Large Language Models (LLMs) and benchmark datasets to compare them. While some models dominate these benchmarks, no single model typically achieves the best accuracy in all tasks and use cases. In this work, we address the challenge of selecting the best LLM out of a collection of models for new tasks. We propose a new formulation for the problem, in which benchmark datasets are repurposed to learn a "router" model for this LLM selection, and we show that this problem can be reduced to a collection of binary classification tasks. We demonstrate the utility and limitations of learning model routers from various benchmark datasets, where we consistently improve performance upon using any single model for all tasks.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated groundbreaking abilities to solve diverse tasks across a variety of NLP domains (Devlin et al., 2018; Brown et al., 2020). Today, researchers in both academia and industry are releasing new LLMs *daily*.[1] These models perform tasks ranging from text classification to question-answering, summarization, and dialogue.

The popularity and influx of open-source LLMs and the diversity of their potential use cases made it crucial to develop comprehensive benchmarks, i.e., collections of datasets representing different tasks and domains to compare LLMs. For example, HELM (Liang et al., 2022) consists of 42 scenarios covering a variety of uses, MMLU (Hendrycks et al., 2020) is a multiple-choice question answering benchmark with 57 tasks organized by topics, Open LLM Leaderboard (Beeching et al., 2023) combines MMLU with other question-answering datasets, and LM Evaluation Harness (Gao et al., 2021) supports over 200 tasks. While there always will be an LLM that is the best *on average* across benchmarks, there is unlikely to ever be a model that is strictly the best *on each* of the hundreds of datasets comprising various benchmarks. Meanwhile, a practitioner typically wants to know what is the best model for their specific use case and is less concerned about average performance on a plethora of other datasets.

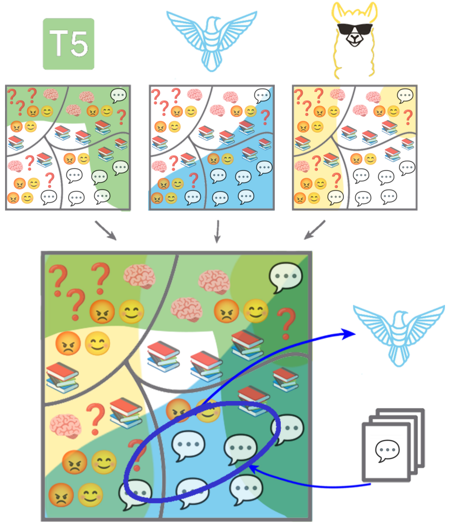


Figure 1: We learn the strengths of candidate LLMs (T5, Falcon, Llama) on various tasks (emojis inside boxes: QA, reasoning, summarization, etc.) and domains (4 sections within each box: finance, legal, general knowledge, etc.) from benchmark datasets. We accomplish this by training a binary classifier per LLM (decision boundaries marked with colors in the upper part of the figure). For a new task (paper stack), we score each LLM with these binary classifiers and recommend an LLM (here Falcon) to the user.

In this paper, we study the problem of identifying the best LLM for a new task. To learn about the strengths and weaknesses of candidate LLMs we use benchmark datasets that give insights into the

[1] Hugging Face currently hosts 22,482 models for text generation.

performance of LLMs across tasks and domains. For example, suppose the new task is answering math questions. In that case, it is more intuitive to consider models that do well on other STEM question-answering datasets and discount performance on, e.g., sociology or toxicity detection. We make this idea precise by casting the learning of model strengths as a binary supervised learning task, where the features are input embeddings of samples across tasks and the labels are whether the model "did well" on the corresponding inputs, e.g., generated correct class label, answered a question correctly, or followed input instructions sufficiently well. See Figure 1 for an illustration. Such information is collected during benchmark evaluations and can be reused for training model routers without having to run expensive LLM inference again. The resulting router is also efficient at test time, as it only requires calling the chosen LLM.

Our contributions are summarized below:

- We formalize the problem of learning the strengths and weaknesses of LLMs for downstream *routing*, i.e., selecting the best model, as a collection of binary classification problems. The goal of each classification problem is to predict whether a given LLM will be "correct" on an input.
- We propose three scores for selecting LLMs for a new task using these correctness predictors. Our third score is designed to account for mistakes a correctness predictor can make on the (out-of-distribution) data from a new task that is likely to be different from the datasets in benchmarks used for training the correctness predictors. We establish connections to meta-learning to obtain theoretical insights into the efficacy of these scores.
- We verify the efficiency of our model routing scores empirically on 29 datasets from HELM (Liang et al., 2022) representing scenarios like question answering, text classification, knowledge, and reasoning, and MixInstruct (Jiang et al., 2023), a collection of datasets for evaluating instruction following capabilities of LLMs.
- We discuss and empirically investigate questions concerning the efficacy and utility of learning LLM routers from benchmarks: generalization of correctness predictors to new tasks, the importance of a larger pool of benchmarks, and the potential of routing smaller LLMs to reduce costs.

2 Related work

Benchmarking Comparing models or algorithms across various tasks is a standard practice in ML and AI literature. Prior to Foundation Models (Bommasani et al., 2021), it was typical to apply *the same learning algorithm* to train a model on each of the datasets and compare the performance against other learning algorithms. The UCI Machine Learning Repository (Kelly et al., 2023) is one prominent example of such a collection of datasets often used to compare learning algorithms. With the emergence of Foundation Models, i.e., models with billions of parameters trained on massive datasets using large compute clusters, the paradigm changed to evaluating *the same model* (or a few-shot tuned version of it) on a variety of tasks (Bojar et al., 2014; Goyal et al., 2019; Li et al., 2022). In the context of Large Language Models, many benchmarks (Wang et al., 2018; 2019; Hendrycks et al., 2020; Gao et al., 2021; Srivastava et al., 2022; Liang et al., 2022; Beeching et al., 2023; Jiang et al., 2023) were proposed to help determine the most capable LLM. Benchmarks typically average the performance of models across tasks and provide a final ranking, discarding the rest of the information. In this work, we use the byproducts of benchmark evaluations, i.e., the per-sample performance of various LLMs across tasks, to learn about their individual strengths and identify the best LLM for a new task.

Model selection Selecting the best model, or model selection, is a classical topic in statistics and ML (Bishop & Nasrabadi, 2006; Hastie et al., 2009; Raschka, 2018). However, the typical problem setting is quite different: classical methods like cross-validation aim to estimate the population error of a model trained on samples from the population distribution. In other words, the goal is to find the best model for in-distribution test data, i.e., data sampled from the same distribution as the train data. The notion of "train" data is quite elusive for LLMs, as they are usually trained on massive datasets with trillions of tokens with a simple task of next token prediction (Radford et al., 2019; Brown et al., 2020). However, the tasks we evaluate them on are often more structured, e.g., classification and question-answering, and are specific to domains that may or may not be sufficiently represented in the train data. In addition, techniques like k-fold cross-validation require training the model multiple times, which is infeasible for LLMs.

Out-of-distribution model selection Recognizing the limitations of the model selection methods for in-distribution test data (Gulrajani & Lopez-Paz, 2021; Koh et al., 2021), recent work has proposed a variety of methods to select models when deployed on data that may differ from the train data. These methods rely on ideas such as bootstrapping (Xu & Tibshirani, 2022), reweighing (Chen et al., 2021b; Maity et al., 2023), agreement of models or ensembles (Jiang et al., 2021; Chen et al., 2021a; Ng et al., 2023), or aligning model accuracy in-distribution with a confidence threshold (Guillory et al., 2021; Garg et al., 2022; Yu et al., 2022). Most of these methods are nontrivial to extend to generation use-cases of LLMs; some require training multiple models, and some need well-defined in-distribution data related to the new task.

Routing LLMs Prior work on selecting LLMs primarily considers choosing one that produces the best generation for a given input. Liu & Liu (2021); Ravaut et al. (2022); Jiang et al. (2023) train dedicated scoring or ranking models that can be applied to model generations. Unlike our work, these approaches require generating outputs with *every* candidate LLM to make a decision, which can be computationally prohibitive with a large pool of candidate LLMs. FrugalGPT (Chen et al., 2023) calls LLMs sequentially until a dedicated scoring model deems the generation acceptable. Prior works in this group require training data sufficiently representative of each of the tasks and domains of interest to train the corresponding ranking and scoring models. Automatic evaluation using other LLMs (Fu et al., 2023; Wang et al., 2023; Liu et al., 2023) may be used instead of such scoring models, but such an approach would require producing generations with every candidate LLM *and* evaluating each of these generations with a typically larger/commercial LLM. In this paper, instead, we use data from benchmarks to learn the strengths and weaknesses of LLMs across tasks and domains. The resulting model router requires generating outputs *only* with the chosen LLM at test time, making the proposed method a lot more cost-efficient.

3 Learning from Benchmarks

We start by introducing notation to describe the majority of NLP benchmarks. Let $\{x_1^d, \dots, x_{n_d}^d\}_{d=1}^D$ be a collection of inputs across D tasks. Each input text x_i^d corresponds to a reference answer r_i^d, i.e., an ideal generation for the corresponding input. Finally, there is a metric $F_d(x, o, r)$ that can be task-dependent and measures how well a response o for an input x corresponds to the reference r. To test an LLM$_m$, $m \in \{1, \dots, M\}$, on the benchmark, for each task $d = 1, \dots, D$, its responses are generated $\{o_{im}^d = \text{LLM}_m(x_i^d)\}_{i=1}^{n_d}$ and compared to the corresponding references to obtain performance metrics $\{f_{im}^d = F_d(x_i^d, o_{im}^d, r_i^d)\}_{i=1}^{n_d}$.[2] At this point, the majority of the benchmark studies will take a (weighted) average of the performance metrics and report a single score for every LLM to rank them in performance. Instead, we reuse these evaluation results to formulate a supervised learning problem to better understand the strengths and weaknesses of various LLMs based on their performance on data points and tasks.

Supervised learning from benchmarks Our goal is to learn a simple routing function $g_m(x)$ for each LLM, $m = 1, \dots, M$, that can predict $\{f_{im}^{d'}\}_{i=1}^{n_{d'}}$, i.e., the performance of the corresponding LLM on a new task d'. Then it is trivial to select the best LLM for this task. For efficiency at test time, we restrict the routers $\{g_m\}_{m=1}^M$ to only depend on the input x. This is in contrast to the majority of prior works on LLM routing that first obtain generations with every candidate LLM and then use them to choose the best model (Liu & Liu, 2021; Ravaut et al., 2022; Jiang et al., 2023). With thousands of open-source LLMs, it is simply infeasible to obtain generations with every LLM for every input at test time.

To complete the problem formulation, we denote the "correctness" of model m on an input x by $y(x, m) \in \{0, 1\}$. Correctness is evaluated as follows: generate a response o_{im}^d with LLM m on input x_i^d, compare it to the corresponding reference r_i^d, and output 1 if the model's response is good enough, i.e., $f_{im}^d > \eta_d$, and 0 otherwise, where η_d is some threshold that can be task and/or metric specific. For tasks like classification or multiple-choice QA, $y(x_i^d, m) = f_{im}^d$, while for various evaluation metrics used in summarization and instruction following tasks (Zhang et al., 2020; Sellam et al., 2020; Yuan et al., 2021), the notion of correctness can help to account for the heterogeneity

[2] We omit dependency on the prompt when generating with an LLM and, w.l.o.g., consider the same LLM with a different prompting strategy as a different LLM.

of popular metrics and task difficulty levels. In Section 5.2, we also present results with raw metrics instead of correctness.

To train a predictor of an LLM correctness, for each LLM, $m = 1, \dots, M$, we solve the following optimization problem:

$$\arg\min_{g_m} \sum_{d=1}^{D} \sum_{i=1}^{n_d} \ell(g_m(x_i^d), y(x_i^d, m)), \tag{1}$$

where we choose ℓ to be a binary cross-entropy loss and g_m is any standard probabilistic classifier, i.e., $g_m(x)$ estimates $P(y(x,m) = 1|x)$. In other words, training g_m is equivalent to simply training a (probabilistic) binary predictor.

An important consideration when training correctness predictors is their ability to generalize out-of-distribution (OOD) data, since our goal is to estimate LLM performance on a new task d' that has not been seen during training. Training predictors given data from multiple domains that need to generalize to unseen domains is indeed an active area of research in ML literature. For example, Sun & Saenko (2016); Arjovsky et al. (2019) proposed methods for improving OOD generalization when training on data from multiple domains, while Koh et al. (2021) proposed a benchmark for OOD generalization demonstrating the challenging nature of the problem in various applications.

In this work, we use a simple model for the correctness predictor: we embed all inputs with a sentence transformer (Reimers & Gurevych, 2019) and use a k-nearest neighbors classifier (Cover & Hart, 1967) as $\{g_m\}_{m=1}^M$. kNN is a simple non-parametric classifier that allows us to fit a potentially complicated decision boundary of an LLM's correctness across multiple tasks without extensive hyperparameter tuning. We choose this approach for learning correctness predictors to emphasize the utility of learning from benchmarks even with a basic method and instead focus on the question specific to our problem that has not been studied in prior works on OOD generalization: *Can we improve the quality of LLM routing with an imperfect correctness predictor*?

4 LLM ROUTING WITH (IMPERFECT) CORRECTNESS PREDICTORS

The goal of LLM routing is to identify an LLM that will have the highest frequency of being correct on a new task d', given the inputs $\{x_i^{d'}\}_{i=1}^{n_{d'}}$ from this task:

$$\arg\max_m \tilde{S}(m, d'), \text{ where } \tilde{S}(m, d') = \tfrac{1}{n^{d'}} \textstyle\sum_{i=1}^{n_{d'}} y(x_i^{d'}, m). \tag{2}$$

Here, $\tilde{S}(m, d')$ is the "oracle" score that we want to estimate. The most intuitive estimator is simply using the correctness predictor

$$S_1(m, d') = \tfrac{1}{n^{d'}} \textstyle\sum_{i=1}^{n_{d'}} g_m(x_i^{d'}), \tag{3}$$

but prior work has shown that accurately estimating $P(y(x,m) = 1|x)$, i.e., training calibrated correctness predictors g_ms, is challenging on OOD data (Ovadia et al., 2019). Meanwhile, g_m may still produce accurate predictions after thresholding the predicted probability even if the class probabilities are not estimated well, which is often the case with neural networks (Guo et al., 2017). This motivates another score:

$$S_2(m, d') = \tfrac{1}{n^{d'}} \textstyle\sum_{i=1}^{n_{d'}} \bar{g}_m(x_i^{d'}), \text{ where } \bar{g}_m(x_i^{d'}) = \mathbb{I}(g_m(x_i^{d'}) > t), \tag{4}$$

where $t \in (0,1)$ is some threshold, e.g., $t = 0.5$, $\mathbb{I}$ is an indicator function, and $\bar{g}_m(x) \in \{0,1\}$ can be interpreted as the prediction of g_m on x. This score, however, does not take into account the potential "imperfection" of g_m, i.e., g_m, as a binary classifier, is likely to have lower accuracy on OOD data from task d'. To address this issue, we model the out-of-distribution confidence of the predictions $\bar{g}_m$.

A simple OOD confidence model We model LLM correctness as follows:

$$y(x,m)|x, d' = \begin{cases} \bar{g}_m(x) & \text{with probability } p(d', m) \\ 1 - \bar{g}_m(x) & \text{with probability } 1 - p(d', m), \end{cases} \tag{5}$$

i.e., $p(d', m) \in [0,1]$ is the probability that $\bar{g}_m$ is the correct prediction on a data point from task d'. The above model can be condensed as follows:

$$y(x,m)|x, d' \sim \text{Bern}(\bar{g}_m(x)p(d', m) + (1 - \bar{g}_m(x))(1 - p(d', m))). \tag{6}$$

In this simplistic (and approximate) model, we assume that $p(d', m)$ does not depend on the input x after conditioning on the task d'. The assumption is analogous to the homoscedastic error term assumption in linear regression models and allows us to interpret $p(d', m)$ as the marginal/overall accuracy of $\bar{g}_m$ on data from the task d'.

Prior work has studied the problem of estimating OOD accuracy given the inputs from a new task, but existing methods are challenging to combine with our approach. For example, Garg et al. (2022) learn a threshold on model confidence, which is hard to apply when using kNN classifiers, and Ng et al. (2023) require data augmentations that can be challenging to identify given the diversity of tasks in benchmarks. Prior methods also do not take into account the partition of the train data into tasks inherent in our problem setting.

We treat the problem of estimating $p(d', m)$ as a supervised learning task, taking advantage of the task partition. Specifically, we assign a task descriptor $u(d) \in \mathbb{R}_+$ to every task that measures the distance of the data from task d to the other available tasks combined. Then we collect the values of $p(d, m)$, i.e., the accuracy of $\bar{g}_m$ on d, and fit a non-parametric regression model to predict $p(d, m)$ from $u(d)$. At test time, we compute $u(d')$ for a new task d' based on the inputs $\{x_i^{d'}\}_{i=1}^{n_{d'}}$ and predict $p(d', m)$ using the fitted regression model. In general, one can consider more sophisticated, higher-dimensional task descriptors $u(d)$, but here, for simplicity, we keep it 1-dimensional and use a Gaussian kernel smoother (also known as the Nadaraya-Watson estimator) as the non-parametric regressor. We provide details in Appendix A.

Finally, given the model of LLM correctness 6, $\tilde{\mathbf{S}}(m, d')$ is a random variable (corresponding to $\tilde{S}(m, d')$) distributed as a (scaled) sum of two Bernoulli random variables. To arrive at our final score for LLM routing, we take its expected value:

$$S_3(m, d') = S_2(m, d')p(d', m) + (1 - S_2(m, d'))(1 - p(d', m)). \tag{7}$$

When selecting an LLM with S_3, we consider an alternative to the $\arg\max$ criterion based on our correctness model 6, which defaults to the best model on average across benchmark datasets when we are not sufficiently confident that a candidate model will be better:

$$\begin{cases} m_3 & \text{if } P(\tilde{\mathbf{S}}(m_3, d') > \tilde{\mathbf{S}}(m^*, d')) > \eta \\ m^* & \text{otherwise,} \end{cases} \tag{8}$$

where $m_3 = \arg\max_m S_3(m, d')$, i.e., the best LLM for the new task according to S_3, and $m^* = \arg\max_m \sum_{d=1}^{D} \tilde{S}(m, d)$, i.e., the best LLM across the benchmark datasets. In the experiments, we set $\eta = 0.6$.

We summarize our LLM routing procedures in Appendix A.

4.1 Connection to meta-learning

The OOD confidence model in equation 6 is a meta-model of routing across multiple tasks, and fitting it entails a form of meta-learning. Consider the meta-learning problem

$$\min_{g_m, p(\cdot, m)} \textstyle\sum_{d=1}^{D} \sum_{i=1}^{n_d} \ell(\bar{g}_m(x_i^d)p(d, m) + (1 - \bar{g}_m(x_i^d))(1 - p(d, m)), y(x_i^d, m)), \tag{9}$$

where $\bar{g}_m$ and $p(\cdot, m)$ are meta-parameters and adaptation step $\bar{g}_m \to \bar{g}_m(\cdot)p(\cdot, m)$ adaptively shrinks the router output towards ambiguity. We exploit this connection to theoretically demonstrate the potential advantages of routing LLMs using S_3 over S_2.

In expectation/in the population, equation 9 fits a larger model class than equation 1, so the risk of the adaptively shrunken router is at most that of the non-adaptive router:

$$\begin{aligned} &\textstyle\sum_{d=1}^{D} \mathbf{E}\big[\ell(\bar{g}_m(X^d)p(d, m) + (1 - \bar{g}_m(X^d))(1 - p(d, m)), y(X^d, m))\big] \\ &\quad \leq \textstyle\sum_{d=1}^{D} \mathbf{E}\big[\ell(\bar{g}_m(X^d), y(X^d, m))\big]. \end{aligned} \tag{10}$$

This suggests (subject to standard assumptions on the loss function) that adaptive shrinkage routing leads to better approximations of the oracle router. Lemma 4.1 confirms this intuition.

Lemma 4.1. *Let $\ell(y_1, y_2) = \rho(y_1 - y_2)$ for some subadditive $\rho : \mathbf{R} \to \mathbf{R}$ (e.g. $\rho(x) = \frac{1}{2}x^2$ for the square loss). We have*

$$\begin{aligned} &\ell(S_2, \widetilde{S}) \leq \mathbf{E}\big[\ell(\bar{g}_m(X^d), y(X^d, m))\big]), \\ &\ell(S_3, \widetilde{S}) \leq \mathbf{E}\big[\ell(p(d, m)\bar{g}_m(X^d) + (1 - p(d, m))(1 - \bar{g}_m(X^d)), y(X^d, m))\big]). \end{aligned}$$

Table 1: LLM routing on HELM: Comparison of various model scores for LLM routing with the Oracle model selection and performance of the best model on average (BMA).

	Acc.	Ratio to Best	Pearson	Spearman	% BMA	# Params	Rank
S_1 eq. 3	0.662	0.855	0.685	0.465	0.17	40.3B	6.172
S_2 eq. 4	0.676	0.868	0.636	0.468	0.10	44.3B	5.897
S_3 eq. 7, 8	0.694	0.898	0.727	0.492	0.48	49.8B	5.310
S_3 true p	**0.735**	**0.944**	**0.799**	**0.596**	0.22	**33.8B**	**3.800**
LL	0.684	0.869	0.714	0.459	0.10	—	6.517
MoE	0.635	0.825	—	—	0.08	34.0B	—
BMA	0.688	0.884	—	—	1.00	70.0B	6.069
Oracle	0.773	1.000	—	—	0.21	29.1B	1.000

We present the proof in Appendix D. Combining equation 10 and Lemma 4.1, we expect the adaptive router based on S_3 to outperform its non-adaptive counterpart based on S_2. That said, it is unclear whether adaptive shrinkage will improve the performance of the adaptive router in finite samples: the expected performance of the adaptive router may be offset by the inflation in variance from fitting the larger (adaptive) model class. Fortunately, our empirical results show that task-specific adaption, i.e., using S_3 as a score for routing, generally improves performance. The two-step method for fitting $\bar{g}_m$ and p in Section 4 approximately minimizes equation 9 with a single Gauss-Seidel pass through the decision variables.

5 Experiments

5.1 Model routing on HELM

Data. We select 29 datasets from the HELM benchmark (Liang et al., 2022) representing scenarios such as question answering, text classification, language, knowledge, and reasoning, among others. We present additional information about these datasets in Table 4.

Models. We evaluate 18 open-source models ranging in size from 3B to 70B, including base and chat variations of Llama 2 in different sizes. All models are summarized in Table 5.

Model routing. The best model on average (BMA) across the 29 considered HELM datasets is `llama-2-70b` (followed by `llama-2-70b-chat`). Our goal is to show that learning model routers from benchmark data can simultaneously outperform BMA and reduce inference costs by recommending smaller LLMs for tasks where they can perform well. We compare models selected with the three scores, S_1, S_2, and S_3, presented in Section 4 to the performance of `llama-2-70b`, i.e., the BMA. All correctness predictors g_ms are kNN classifiers with $k = 5$. We also consider Mixture of Experts (MoE) (Masoudnia & Ebrahimpour, 2014) as a baseline where an LLM is selected for each input individually based on the correctness frequency of candidate LLMs on the same k neighbors of this input. Results for neural networks as correctness predictors are in Appendix B.1.

We also report the performance of the best model according to the "oracle" score $\tilde{S}$, which is the upper bound on what can be achieved with model routing, and $\tilde{S}_3$, which corresponds to S_3 with the true $p(d', m)$, i.e., the accuracy of (an imperfect) g_m on d'. Finally, we compare the scoring LLMs with the average log-likelihood (LL) (or negative perplexity) of the response they generate on the inputs from the task of interest. This last baseline requires producing generations with *every* LLM at test time to make a selection, while all of our scores only require generating with the chosen LLM.

Results. We conduct 29 sets of experiments, each time selecting 28 of the datasets as the benchmark data for training the LLM routers and using the remaining task as the new task d' for evaluating the quality of the LLM selection for this task. In Table 1 we report averages across experiments for the performance of the selected model (Acc.), ratio of this performance to the performance of the best model for the corresponding new task (Ratio to Best), Pearson and Spearman rank correlations between model accuracies and model scores, number of parameters of the selected model (# Params), rank of the selected model out of 18 considered (Rank). We also report the fraction of times the BMA is selected by a method (% BMA). Best results are highlighted with bold and second best with an underline (excluding Oracle).

First, we notice that accounting for imperfections of the correctness predictors (their average accuracy is 0.59) has clear benefits: when we have access to the true accuracy of correctness predictors, the corresponding score, S_3 true p, noticeably outperforms all other scores. Our simple kernel smoothing estimator of this accuracy (MAE= 0.116) allows us to obtain a practical model routing score S_3 that outperforms BMA (`llama-2-70b`) while choosing smaller models for some of the tasks (as evident by the average number of parameters of the chosen models). S_2 sacrifices some accuracy but chooses even smaller performant models. Overall, learning from benchmarks allows us to obtain LLM routers that can improve overall performance while utilizing smaller models where appropriate. Finally, we note that log-likelihood (LL) also performs well. However, routing with it requires passing each test input through *each* candidate LLM, which has 347B parameters in total.

Reducing the OOD gap. The average accuracy of correctness predictors across tasks and models for the experiments in Table 1 is 0.59. It is a fairly low accuracy for binary classification, which we attribute to the diversity of tasks in HELM leading to substantial distribution shifts when predicting the correctness of LLMs on held-out tasks. We investigate the quality of model routing when we reduce this OOD gap. A simple strategy to reduce this gap is to collect a small number of labeled in-distribution samples. This can be accomplished by asking a practitioner to provide reference answers ($r_i^{d'}$s) for a small number of inputs from their task to evaluate the correctness of candidate LLMs on these in-distribution inputs and use it to improve correctness predictors.

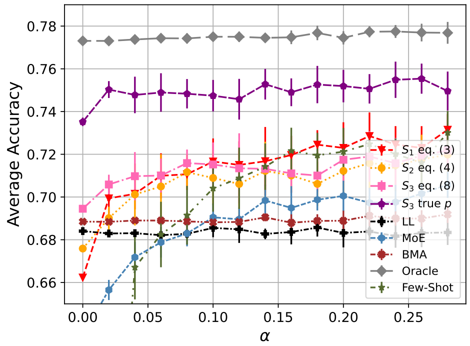


Figure 2: Using $\min(\alpha n^{d'}, 50)$ training samples from d' to reduce OOD gap.

We simulate this scenario by moving $\min(\alpha n^{d'}, 50)$ samples from the data from a new task d' to the data for training the correctness predictors. The upper limit of 50 samples is to maintain practical utility while accounting for varying dataset sizes (see Table 4). In this experiment, we additionally compare to Few-Shot, which selects an LLM for the new task based on the performance only on these labeled samples from the new task. We conduct 29 sets of experiments, repeating each one 10 times to obtain standard deviations (randomness is due to random selection of data points from a new task for reducing the OOD gap). We summarize the average accuracy of models selected with various routing scores for varying α in Figure 2 ($\alpha = 0$ corresponds to Table 1). Results for Pearson correlation are in Figure 6(a).

We see that even a small number of in-distribution samples ($\alpha = 0.05$) can reduce the OOD gap (corresponding average accuracy of correctness predictors is 0.65; see Figure 6(b)) and noticeably improves the model routing performance of all three of our scores. When the number of in-distribution samples further increases, S_1 starts to outperform S_3. We attribute this observation to kNN being well-calibrated in-distribution, i.e., the correctness predictors provide reliable estimates of their own confidence $P(y(x, m) = 1|x)$, which are used by S_1 in equation 3. Finally, we note a fairly large variance in the results due to random selection of the in-distribution training samples from d', suggesting that active learning (Settles, 2009) can help to improve LLM routing further.

5.2 Model Routing on Mix-Instruct

We further demonstrate our approach in a different setting and task type, on the MixInstruct benchmark dataset (Jiang et al., 2023). The dataset is composed of instruction-following tasks, divided into train/validation/test sets of 100K/5K/5K samples, and includes evaluations of $N = 11$ open-source LLMs using common metrics, e.g. BERTScore (Zhang et al., 2020), BARTScore (Yuan et al., 2021), and BLEURT (Sellam et al., 2020). In Jiang et al. (2023), this benchmark was used to compare different LLM ranking methods in per-instance model selection. We follow the same setting and apply our score $S_1(m, d')$ to the test set, per-instance, where we use the 100K-sample train set as the benchmark data for training our LLM router. See Appendices A and C for details on the score computation and the experiment parameters, respectively. Due to the per-instance setting and the test set constructed from in-distribution data, we focus on our simplest router model S_1, equation 3.

We compare our approach with the scoring methods examined by Jiang et al. (2023), as well as scoring based on the average log-likelihood (LL) of the model responses to the inputs. Additionally,

Table 2: Average metrics for per-instance LLM selection on the MixInstruct test set. Best results are highlighted with bold and second best with an underline (excluding Oracle).

	BERTScore ↑	BARTScore ↑	BLEURT ↑	MCPI
Random	66.36	-3.76	-0.77	-
LL	65.83	-4.12	-0.96	N
BMA: Open-Assisant	<u>74.68</u>	-3.45	-0.39	-
BMA: Vicuna	69.60	-3.44	-0.61	-
MLM-Scoring (Salazar et al., 2020)	64.77	-4.03	-0.88	N
SimCLS (Liu & Liu, 2021)	73.14	<u>-3.22</u>	<u>-0.38</u>	N
SummaReranker (Ravaut et al., 2022)	71.60	-3.25	-0.41	N
PairRanker (Jiang et al., 2023)	72.97	**-3.14**	**-0.37**	N
Ours	**74.75**	-3.40	<u>-0.38</u>	**2**
Oracle	77.67	-2.87	-0.15	N

we present the metrics for the best models on average (BMA), Open-Assistant (LAION-AI, 2023) and Vicuna (Chiang et al., 2023).

We report the results of BERTScore, BARTScore and BLEURT in Table 2, along with the number of model calls per instance (MCPI), for N LLMs, performed during inference time. All compared methods require model generations for every point in the test set, by each of the examined LLMs, whereas our approach requires only one model generation and one call to some general embedding function. In addition, all methods, except for LL, require training auxiliary language models, whereas our approach is a simple kNN classifier on the embedded inputs. While our approach does not consistently outperform the compared methods, these results demonstrate the potential of using benchmark datasets for model routing with significantly better inference-time efficiency.

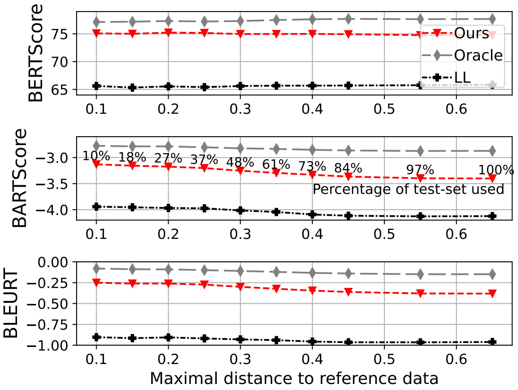


Figure 3: Average metrics on subsets of the MixInstruct test set, defined by limiting the maximal average distance between test instances and their closest neighbors in the reference (train) set.

Effect of benchmark dataset sparsity. To highlight the potential of our approach in this setting, we examine the effect of the reference benchmark data sparsity. We apply our method to different subsets of the test set, X_{test}, where the subsets are defined by limiting the maximal average distance of each test set point to the closest points from the reference (train) set, denoted by NN_{train}, i.e. $X'_C = \left\{ x' \in X_{\text{test}} \middle| \frac{1}{|\text{NN}_{\text{train}}(x')|} \sum_{x \in \text{NN}_{\text{train}}(x')} \text{dist}(x', x) < C \right\}$, where C is the maximal average distance and X'_C is the resulting subset of the test set. Figure 3 presents the metric scores for the different subsets using our method, the oracle (best possible choices), and LL scoring. We also report the percentage of the test set that is used in each subset. This figure depicts that our predictor approaches the oracle metrics as the average distance to the reference points decreases. This suggests that adding more benchmark datasets to reduce the sparsity of the reference space may lead to better LLM selections with our approach.

6 Discussion and Conclusion

How useful are smaller LLMs? While a given LLM may work best on average, these models tend to be the biggest and therefore most expensive to run. Practitioners can achieve gains in cost, compute, and latency if we can successfully predict whether a smaller LLM can be adequate for a given task. Identifying good smaller models for tasks of interest will also redefine the cost/benefit tradeoff behind automating certain tasks, potentially incentivizing the automation of new tasks that were previously cost-prohibitive to automate with larger LLMs.

To evaluate the potential of smaller LLMs we revisit our HELM experiment in Figure 2. In Figure 4, we perform LLM routing using *only models with* $\leq$ *13B parameters* and compare it to the performance of `Llama 2 70B`. Oracle's performance demonstrates that it is conceptually possible to outperform a large model by routing smaller LLMs. Results with our scores S_1 and S_2 (see Figure 7 for breakdown by scores) demonstrate that it is also practically feasible to match the performance of the 70B model by combining learning from benchmarks with a small number ($\alpha = 0.04$, i.e., 2-40 samples) of labeled samples from a new task that a practitioner can provide to save on the inference costs in their LLM application.

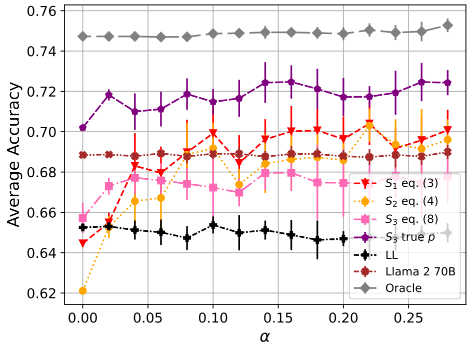


Figure 4: LLM routing with $\leq$ 13B parameter models compared to `Llama 2 70B`.

Learning from more benchmarks. We anticipate learning LLM routers from benchmarks to be the most effective when new tasks are similar to the benchmark tasks, thus reducing the OOD gap without any labeling burden for a practitioner. To empirically investigate this hypothesis, in Figure 5 we visualize the relation between the quality of model routing with S_3, measured with Pearson correlation between model scores and accuracies of candidate LLMs, and the distance $u(d')$ from a new task d' to the available benchmark data for training the routers. In this experiment, we aggregate results across different α values from Figure 2. For smaller distance values the correlation is approaching 1, while for large distances it sometimes deteriorates. Results for other scores demonstrate a similar trend and are presented in Appendix B.3 along with additional details. This experiment and the benchmark dataset sparsity analysis presented in Figure 3 for MixInstruct illustrate that learning with *more benchmarks* can improve the efficacy and reliability of the LLM routers as new tasks are more likely to be closer to a large collection of datasets.

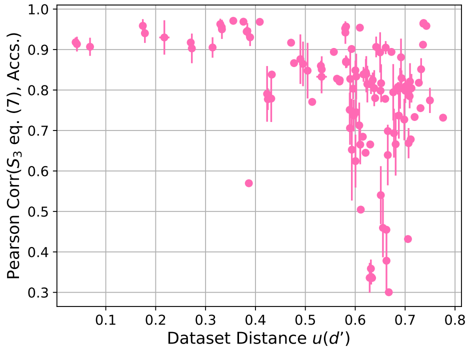


Figure 5: Correlation(S_3, Accs.) and $u(d')$.

Future work. Our work demonstrates the potential of learning from benchmarks for LLM routing and investigates 3 model scores in the context of OOD generalization when routing LLMs for new tasks. We summarize potential next steps for improving the quality and efficacy of LLM routers.

The major challenge of LLM routing is OOD generalization of correctness predictors. Thus, using more benchmarks and modern methods for improving OOD generalization to learn correctness predictors is a promising next step. A practitioner can also provide labels for a few samples from their task, possibly guided by active learning techniques, to adapt or fine-tune correctness predictors. Even when reducing the OOD gap is too challenging, our score accounting for the (potentially low) accuracy of correctness predictors demonstrated strong results when this accuracy, $p(d', m)$, is known for a new task, thus encouraging the development of methods for estimating it better.

We also anticipate that routing "expert" LLMs fine-tuned for a specific domain can improve the results. Regions of the sample space where such models are "correct" should mostly align with the domains of their expertise (recall Figure 1), making it easier to learn the corresponding correctness predictors and simplifying LLM routing when a new task is from a specific domain.

Our experiments in Figure 4 demonstrate the utility of LLM routing with *smaller* models, which can reduce costs and facilitate using LLMs in a broader set of domains. Thus, we want to explore modifications to our scores that will encourage the selection of smaller LLMs when their anticipated performance is comparable to the larger, more reliable models. Prior work on frugal API selection (Chen et al., 2020; 2023) provides a good starting point to explore this direction.

Finally, it is interesting to explore other approaches to correctness prediction. For example, Mixture of Experts (Masoudnia & Ebrahimpour, 2014) perspective suggests training a single model to identify the best LLM for each input instead of predicting the correctness of each LLM individually.

References

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard. https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58, 2014.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

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

Jiefeng Chen, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. *Advances in Neural Information Processing Systems*, 34:14980–14992, 2021a.

Lingjiao Chen, Matei Zaharia, and James Y Zou. Frugalml: How to use ml prediction apis more accurately and cheaply. *Advances in neural information processing systems*, 33:10685–10696, 2020.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *arXiv preprint arXiv:2305.05176*, 2023.

Mayee Chen, Karan Goel, Nimit S Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Ré. Mandoline: Model evaluation under distribution shift. In *International conference on machine learning*, pp. 1617–1629. PMLR, 2021b.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*, 2023.

Leo Gao, Jonathan Tow, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Kyle McDonell, Niklas Muennighoff, Jason Phang, Laria Reynolds, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Saurabh Garg, Sivaraman Balakrishnan, Zachary Chase Lipton, Behnam Neyshabur, and Hanie Sedghi. Leveraging unlabeled data to predict out-of-distribution performance. In *International Conference on Learning Representations*, 2022.

Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the ieee/cvf International Conference on computer vision*, pp. 6391–6400, 2019.

Devin Guillory, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. Predicting with confidence on unseen distributions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1134–1144, 2021.

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pp. 1321–1330. PMLR, 2017.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, jul 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792.

Yiding Jiang, Vaishnavh Nagarajan, Christina Baek, and J Zico Kolter. Assessing Generalization of SGD via Disagreement. In *International Conference on Learning Representations*, 2021.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI Machine Learning Repository, 2023. URL https://archive.ics.uci.edu.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

LAION-AI. Open assistant, 2023. URL https://github.com/LAION-AI/Open-Assistant.

Chunyuan Li, Haotian Liu, Liunian Li, Pengchuan Zhang, Jyoti Aneja, Jianwei Yang, Ping Jin, Houdong Hu, Zicheng Liu, Yong Jae Lee, et al. Elevater: A benchmark and toolkit for evaluating language-augmented visual models. *Advances in Neural Information Processing Systems*, 35: 9287–9301, 2022.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*, 2023.

Yixin Liu and Pengfei Liu. Simcls: A simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 1065–1072, 2021.

Subha Maity, Mikhail Yurochkin, Moulinath Banerjee, and Yuekai Sun. Understanding new tasks through the lens of training data via exponential tilting. In *International Conference on Learning Representations*, 2023.

Saeed Masoudnia and Reza Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.

Nathan Hoyen Ng, Neha Hulkund, Kyunghyun Cho, and Marzyeh Ghassemi. Predicting out-of-domain generalization with neighborhood invariance. *Transactions on Machine Learning Research*, 2023.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

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

Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. Summareranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4504–4524, 2022.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2019.

Julian Salazar, Davis Liang, Toan Q Nguyen, and Katrin Kirchhoff. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2699–2712, 2020.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7881–7892, 2020.

Burr Settles. Active learning literature survey. 2009.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pp. 443–450. Springer, 2016.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*, 2023.

Hui Xu and Robert Tibshirani. Estimation of prediction error with known covariate shift. *arXiv preprint arXiv:2205.01849*, 2022.

Yaodong Yu, Zitong Yang, Alexander Wei, Yi Ma, and Jacob Steinhardt. Predicting out-of-distribution error with the projection norm. In *International Conference on Machine Learning*, pp. 25721–25746. PMLR, 2022.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277, 2021.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*, 2020.

A Correctness predictors and confidence estimation

We provide additional details on the correctness predictors used in our experiments, along with more details on the dataset distance and the Gaussian kernel smoother for estimating the accuracy of the correctness predictors on new tasks, $p(d', m)$s.

Correctness predictors in our experiments While any probabilistic classifier may fit our setting, in the experiments, we mainly used a simple kNN classifier applied in an embedded space. Recall that we have D benchmark datasets with inputs $\{x_i^d\}_{i=1}^{n_d}$ for $d = 1, \ldots, D$. To compute our correctness predictor based on the benchmark datasets, we first embed all their inputs. We denote the combined set of embedded inputs from the benchmark datasets as $\mathcal{D} = \{\phi(x_1^d), \ldots, \phi(x_{n_d}^d)\}_{d=1}^D$, where ϕ is a sentence transformer (Reimers & Gurevych, 2019). We use `all-mpnet-base-v2` from Hugging Face in all experiments. Given a sample $x_i^{d'}$ from a new task d', we embed it using the same ϕ and define the classifier, g_m, for each model m by:

$$g_m\left(x_i^{d'}\right) = \frac{1}{k} \sum_{e \in \text{NN}(\phi(x_i^{d'}), k, \mathcal{D})} y(e, m), \tag{11}$$

where $y(e, m) \in \{0, 1\}$ is the correctness of model m on the (embedded) input e, and $\text{NN}(\phi(x_i^{d'}), k, \mathcal{D})$ is the set of k closest embedded neighbors from $\mathcal{D}$ to the new embedded sample $\phi(x_i^{d'})$, according to the cosine distance. Then, $\bar{g}_m$, as defined in equation 4, is a binary kNN classifier. Finally, we compute the per-model correctness predictors, $S_1(m, d')$ and $S_2(m, d')$, for the new task d', according to equation 3 and equation 4, respectively.

Next, we describe a method for estimating the probability $p(d', m)$ in our confidence model and the $S_3(m, d')$ score, equation 7. This method comprises a task descriptor based on dataset distance and a kernel smoother, defined as follows.

Task descriptor as dataset distance To compare tasks we need to assign them some numerical representations, i.e., task descriptors. Here we utilize dataset distance as task descriptors as follows. Our dataset distance $u(d)$ is a one-sided variant of the Chamfer distance with extended neighborhood size. We define it formally below:

$$u(d) = \frac{1}{n_d} \sum_{i=1}^{n_d} nn(x_i^d, \mathcal{D}_{-d}), \tag{12}$$

where $\mathcal{D}_{-d}$ is the set of (embedded) inputs from the D datasets excluding inputs from d (for a new task d', $\mathcal{D}_{-d'} = \mathcal{D}$ since d' is not part of the D benchmark datasets we use for training LLM routers), and $nn(x_i^d, \mathcal{D}_{-d})$ is the average distance from the input x_i^d to its closest κ neighbors in $\mathcal{D}_{-d}$:

$$nn(x, \mathcal{D}) = \frac{1}{\kappa} \sum_{e \in \text{NN}(\phi(x), \kappa, \mathcal{D})} \text{cosine}(\phi(x), e), \tag{13}$$

where $\text{NN}(\phi(x), \kappa, \mathcal{D})$ is the set of κ closest embedded neighbors of $\phi(x)$ in $\mathcal{D}$ according to cosine distance. We set $\kappa = 19$ for the dataset distance in all experiments.

Kernel smoother For each LLM $m = 1, \ldots, M$, to obtain the corresponding kernel smoother estimate we iterate over the available benchmark datasets, each time holding one out and computing pairs $(u(d), p(d, m))$ for held out dataset d, where $p(d, m)$ is the accuracy of g_m on data from d after training on $\mathcal{D}_{-d}$. We repeat this process 10 times for 15 values of in-distribution mixing parameter α (similar to the experimental setup in Figure 2 but using benchmark datasets $d = 1, \ldots, D$ instead of d') to obtain the training set of distance-accuracy pairs $\{u_z, p_z(m)\}_{z=1}^Z$. In the HELM experiments in Section 5.1, $Z = 28 * 10 * 15 = 4200$ (28 is the number of datasets from HELM after holding one out as the new task for evaluating the performance).

For a new task d', we compute $u(d')$ using the inputs from this task and our benchmark datasets and estimate $p(d', m)$ for each m with simple Gaussian kernel smoothing:

$$p(d', m) = \frac{\sum_{z=1}^Z p_z(m) \mathcal{K}(u(d'), u_z)}{\sum_{z=1}^Z \mathcal{K}(u(d'), u_z)}, \tag{14}$$

Table 3: Average metrics for the experiments using MLP. Best results are highlighted with bold, and second best with an underline (excluding Oracle).

	Acc.	Ratio to Best	Pearson	Spearman	% BMA	# Params	Rank
S_1 eq. 3	0.693	0.893	0.750	0.559	0.19	57.6	4.345
S_2 eq. 4	0.683	0.878	0.675	0.547	0.45	57.9	5.041
S_3 eq. 7, 8	0.690	0.884	0.755	0.544	0.81	66.3	5.566
S_3 ATC	0.646	0.827	0.651	0.365	0.56	**48.7**	7.283
S_3 true p	**0.740**	**0.943**	**0.840**	**0.686**	0.43	50.4	**3.455**
LL	0.684	0.869	0.714	0.459	0.10	—	6.517
BMA	0.688	0.884	—	—	1.00	70.0	6.069
Oracle	0.773	1.000	—	—	0.21	29.1	1.000

where $\mathcal{K}(u(d'), u_z) = \exp\left(-\frac{(u(d')-u_z)^2}{2\sigma^2}\right)$. We set $\sigma = 0.09$ in all experiments, which is the value we found to perform well through some preliminary experimentation.

Finally, we note that the proposed confidence model, including the definitions of the dataset distance and kernel smoother, can be combined with any classifier g_m, and is not restricted to the kNN classifier used for the correctness predictor in our experiments.

Additional notes regarding S_3 Recall that when selecting a model with $S_3(m, d')$ we use an additional step described in equation 8 that facilitates the selection of the best model on average when we are not sufficiently confident in the model with the highest $S_3(m, d')$ score. Probability expression, $P(\tilde{\mathbf{S}}(m_3, d') > \tilde{\mathbf{S}}(m^*, d'))$, required for this step is not available in closed form, as $\tilde{\mathbf{S}}$ is distributed as a (scaled) sum of two Bernoulli random variables, but it is straightforward to estimate via Monte Carlo sampling from the corresponding Bernoulli distributions.

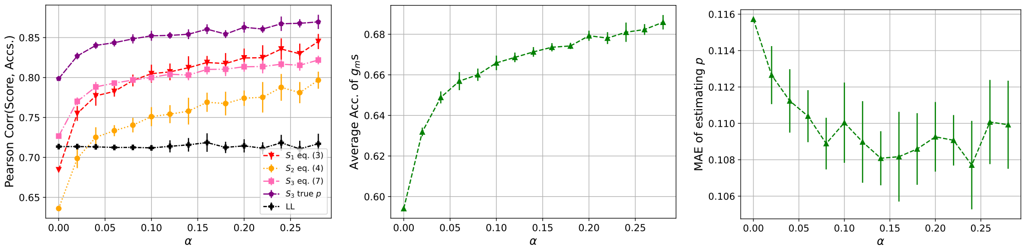
When reporting correlations for S_3 (e.g., Pearson and Spearman correlations in Table 1), we use $S_3(m, d')$ as is, i.e., as defined in equation 7.

B Additional results for model routing on HELM

B.1 Results with MLP as the correctness predictor

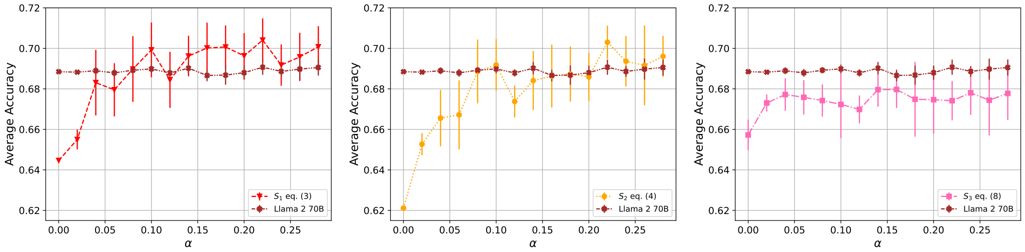
MLP Classifier We present LLM routing with more sophisticated correctness predictors. In this section, we use a small Multi-Layer Perceptron (MLP) as g_m to classify the embedded inputs. Each MLP comprises three fully connected layers (two hidden layers, 1500 units each) with ReLU activation and sigmoid output. These MLPs were trained with the Adam optimizer (Kingma & Ba, 2014) with learning rate 0.01 to minimize the binary cross entropy loss described in equation 1. Training occurred over 100 epochs with early stopping should the validation accuracy not increase for 10 consecutive epochs. We also explore the combination of S_3 with ATC (instead of the kernel smoother), a prior OOD accuracy estimator proposed by Garg et al. (2022).

Results We conducted the model routing experiment analogous to the experiment in Table 1 of the main paper. Results are reported in Table 3. We note several differences compared to results with the kNN classifier in Table 1. First is the clear improvement in the S_1 score; the MLP classifier performs better on OOD data. As before S_3 true p outperforms the other scores indicating the potential value of using the proposed confidence model 6. However, in this experiment, S_3 with the kernel smoother no longer improves upon S_1. We also report results with ATC as the estimator of the accuracy of correctness predictors, which performs noticeably worse than our kernel smoothing estimator. The corresponding MAE is 0.118 for the kernel smoother and 0.177 for the ATC, demonstrating the advantage of our estimator in this application. Finally, across all scores the selected model sizes have considerably increased; this suggests that the MLP classifier underestimates the performance of smaller models compared to kNN.



(a) Pearson correlation (b) Accuracy of g_ms (c) Estimation of $p(d', m)$s

Figure 6: Additional results for Reducing the OOD gap experiment in Figure 2.



(a) S_1 equation 3 (b) S_2 equation 4 (c) S_3 equation 8

Figure 7: LLM routing with $\leq$ 13B parameter models compared to `Llama 2 70B`.

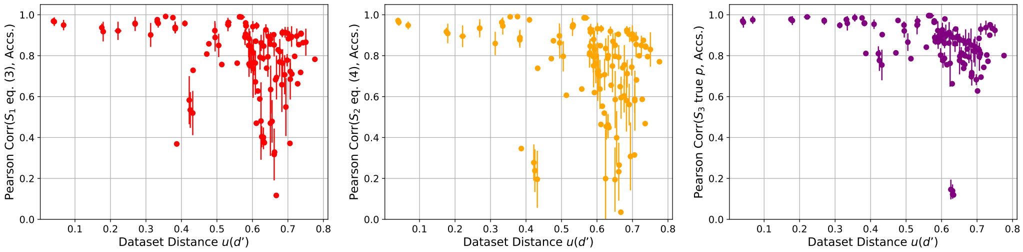
B.2 Reducing the OOD gap

We present additional results for this experiment in Figure 6. (a) shows Pearson correlation improvement as we increase α, similar to the trends in accuracy improvement in Figure 2; (b) demonstrates that the accuracy of correctness predictors g_ms improves as we increase the number of samples from d' used for training them, thus reducing the OOD gap; (c) shows the mean absolute error (MAE) of our kernel smoothing estimator of the accuracy of correctness predictors $p(d', m)$ – the estimator does not improve as much with increased α, thus S_3 eventually becomes worse than S_1 in terms of correlation and accuracy of the selected models.

B.3 Dataset distance and Pearson correlation

The dataset distance $u(d')$ is computed as in equation 12. As evident from equation 13, dataset distance will usually decrease for larger values of α as inputs from d' are moved into $\mathcal{D}$ (assuming that inputs from d' are on average closer to each other than they are to inputs from other tasks). In this experiment, this serves as a mechanism to study the performance of LLM routing on closer datasets, providing insights into the benefits of learning LLM routers on *more benchmarks* where it is more likely that dataset distance for a new task is small.

In Figure 8 we present relations between dataset distance $u(d')$ and Pearson correlation between various model scores and accuracies of candidate LLMs. For results with S_3 see Figure 5.



(a) S_1 equation 3 (b) S_2 equation 4 (c) S_3 equation 7; true $p(d', m)$

Figure 8: Correlation of scores and LLM accuracies on new tasks and corresponding data distances.

C ADDITIONAL DETAILS FOR MODEL ROUTING ON MIXINSTRUCT

Correctness predictor and metrics In the experiments on the MixInstruct dataset (Jiang et al., 2023), we construct $S_1(m, d')$ following the scoring approach described in Appendix A, where the MixInstruct train set was defined as the benchmark dataset. Then, instead of computing a per-dataset score for the entire test set, we compute the score for each test point, i.e., $S_1(m, x_i^{d'}) = g_m(x_i^{d'})$, and select a model m per-point based on this score. The reported metrics in Table 2 and Figure 3 are averaged over the output evaluations of these per-point model selections. In our experiments, to compute $g_m(x_i^{d'})$ we use the BERTScore metric on the closest train set points (as $y(x, m)$ in 11). This was motivated by the conceptual relation between the implementation of our approach and the BERTScore, which relies on embedding space distances, and was validated empirically.

kNN parameter We set $k = 10$ for the kNN classifier, slightly higher than in the HELM experiments. This choice was motivated by the in-distribution properties of the test set in MixInstruct, which is constructed from different parts of the same datasets that comprise the train set. We note that the metrics did not significantly vary for different choices of $k \in [5, 100]$.

D PROOF OF LEMMA 4.1

Lemma D.1 (Lemma 4.1)**.** *Let $\ell(y_1, y_2) = \rho(y_1 - y_2)$ for some subadditive $\rho : \mathbf{R} \to \mathbf{R}$ (e.g. $\rho(x) = \frac{1}{2}x^2$ for the square loss). We have*

$$\begin{aligned}
\ell(S_2, \widetilde{S}) &\leq \mathbf{E}\big[\ell(\bar{g}_m(X^d), y(X^d, m))\big]), \\
\ell(S_3, \widetilde{S}) &\leq \mathbf{E}\big[\ell(p(d, m)\bar{g}_m(X^d) + (1 - p(d, m))(1 - \bar{g}_m(X^d)), y(X^d, m))\big]).
\end{aligned}$$

Proof. We start by showing the upper bound of $\ell(S_2, \widetilde{S})$:

$$\begin{aligned}
\ell(S_2, \widetilde{S}) &= \rho(\mathbf{E}\big[\bar{g}_m(X^d)\big] - \mathbf{E}\big[y(X^d, m)\big]) && \text{(def of } \ell) \\
&= \rho(\mathbf{E}\big[\bar{g}_m(X^d) - y(X^d, m)\big]) \\
&\leq \mathbf{E}\big[\rho(\bar{g}_m(X^d) - y(X^d, m))\big]) && \text{(convexity of } \rho) \\
&= \mathbf{E}\big[\ell(\bar{g}_m(X^d), y(X^d, m))\big]), && \text{(def of } \ell)
\end{aligned}$$

where we recalled that subadditive functions are convex in the third step. The upper bound of $\ell(S_3, \widetilde{S})$ follows a similar argument:

$$\begin{aligned}
\ell(S_3, \widetilde{S}) &= \rho(p(d, m)\mathbf{E}\big[\bar{g}_m(X^d)\big] + (1 - p(d, m))(1 - \mathbf{E}\big[\bar{g}_m(X^d)\big]) - \mathbf{E}\big[y(X^d, m)\big]) \\
&= \rho(\mathbf{E}\big[p(d, m)\bar{g}_m(X^d) + (1 - p(d, m))(1 - \bar{g}_m(X^d)) - y(X^d, m)\big]) \\
&\leq \mathbf{E}\big[\rho(p(d, m)\bar{g}_m(X^d) + (1 - p(d, m))(1 - \bar{g}_m(X^d)) - y(X^d, m))\big]) \\
&= \mathbf{E}\big[\ell(p(d, m)\bar{g}_m(X^d) + (1 - p(d, m))(1 - \bar{g}_m(X^d)), y(X^d, m))\big]) \\
&\leq \mathbf{E}\big[\ell(\bar{g}_m(X^d), y(X^d, m))\big]).
\end{aligned}$$

□

Table 4: HELM dataset details

Dataset	Size (instances)	Type
RAFT-ADE Corpus V2	40	Binary Classification
RAFT-Banking 77	40	77 Class Classification
RAFT-NeurIPS Impact Statement Risks	40	Binary Classification
RAFT-One Stop English	40	3 Class Classification
RAFT-Overruling	40	Binary Classification
RAFT-Semiconductor Org Types	40	3 Class Classification
RAFT-Systematic Review Inclusion	40	Binary Classification
RAFT-TAI Safety Research	40	Binary Classification
RAFT-Terms of Service	40	Binary Classification
RAFT-Tweet Eval Hate	40	Binary Classification
RAFT-Twitter Complaints	40	Binary Classification
IMDB	1000	Binary Classification
Civil Comments-demographic=all	1000	Binary Classification
bAbI-QA-task=all	1000	Q&A: one word answers
BoolQ	1000	Binary Classification
Entity Matching-Dataset=Beer	182	Binary Classification
Entity Matching-Dataset=Dirty iTunes Amazon	218	Binary Classification
Entity Matching-Dataset=Abt Buy	1000	Binary Classification
Entity Data Imputation-Dataset=Restaurant	242	Q&A: one word answers
Entity Data Imputation-Dataset=Buy	182	Q&A: one word answers
BBQ-subject=all	1000	Multiple Choice Questions
Legal Support	1000	Multiple Choice Questions
LSAT QA-task=all	461	Multiple Choice Questions
MMLU-Subject=Abstract Algebra	111	Multiple Choice Questions
MMLU-Subject=College Chemistry	108	Multiple Choice Questions
MMLU-Subject=Computer Security	111	Multiple Choice Questions
MMLU-Subject=Econometrics	126	Multiple Choice Questions
MMLU-Subject=US foreign policy	111	Multiple Choice Questions
Truthful QA-task=mc single	654	Multiple Choice Questions
Total: 29 datasets	9946	

Table 5: Candidate LLMs

Name	Model Size, B	Average Accuracy on the 29 HELM tasks
codegen-16b-mono	16	0.451
dial-flant5-xl	3	0.454
falcon-40b	40	0.641
flan-t5-xl	3	0.650
flan-t5-xxl	11	0.658
flan-ul2	20	0.668
gpt-jt-6b-v1	6	0.576
gpt-neox-20b	20	0.492
mpt-7b-instruct	7	0.514
mt0-xxl	13	0.543
llama-2-13b	13	0.624
llama-2-13b-chat	13	0.623
llama-2-13b-chat-beam	13	0.603
llama-2-70b	70	**0.688**
llama-2-70b-chat	70	<u>0.687</u>
llama-2-7b	7	0.610
llama-2-7b-chat	7	0.605
starcoder	15	0.587
Total: 18 LLMs	347	