Large Language Model Routing with Benchmark Datasets

by

Anthony C Ou

B.S. Electrical Engineering and Computer Science and Physics, Massachusetts Institute of Technology, 2023

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2024

Authored by:	Anthony C Ou Department of Electrical Engineering and Computer Science January 19, 2024
Certified by:	Neil Thompson Research Scientist, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair Master of Engineering Thesis Committee

Large Language Model Routing with Benchmark Datasets

by

Anthony C Ou

Submitted to the Department of Electrical Engineering and Computer Science
on January 19, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

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. With a new dataset, it can be difficult to determine which LLM is best suited to the task. In this work we will address the challenges associated with selecting the best LLM model out of a collection for a new task. To do so, benchmark datasets are repurposed to learn a "router" model for this LLM selection, such that the "router" model will solve a collection of binary classification tasks. This work will demonstrate the utility and limitations of learning model routers from various benchmark datasets, where performance is improved upon using any single model for all tasks.

Thesis supervisor: Neil Thompson
Title: Research Scientist

Acknowledgments

I would like to thank Mikhail Yurochkin for his unwavering support and guidance in all technical matters. I would also like to thank Neil Thompson for his leadership and expertise to guide the direction of research. Much of the work on this thesis was done in collaboration with Tal Shnitzer and Mirian Silva; I am grateful for their contributions to this work. Finally, I would like to express appreciation to the MIT-IBM Watson AI Lab for providing funding and compute resources for this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large Language Models (LLMs) have demonstrated groundbreaking abilities to solve diverse tasks across a variety of NLP domains [1], [2]. Today, researchers in both academia and industry are releasing new LLMs *daily*. 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

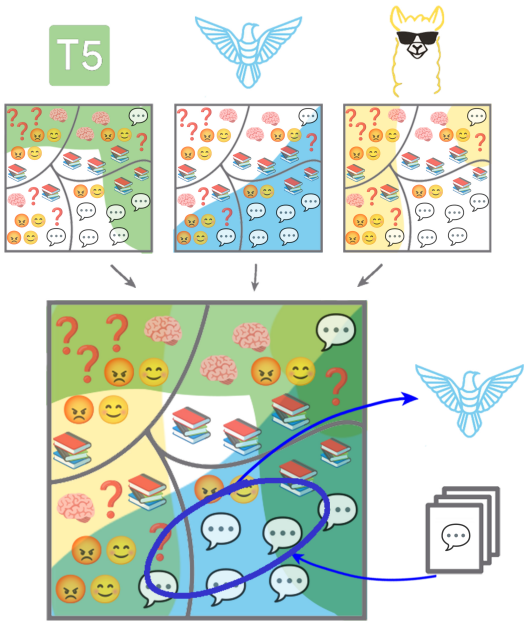


Figure 1.1: Model Routing Schematic: We learn the strengths of candidate LLMs (T5, Falcon, Llama) on various tasks (emojis inside boxes: QA, reasoning, summarization, etc.) and domains (4 sections within each box: finance, legal, general knowledge, etc.) from benchmark datasets. We accomplish this by 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.

representing different tasks and domains to compare LLMs. For example, HELM [3] consists of 42 scenarios covering a variety of uses, MMLU [4] is a multiple-choice question answering benchmark with 57 tasks organized by topics. 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.

We will 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 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. More specifically, we propose a strategy of using the LLM performances on benchmark datasets to predict how it will perform on a new dataset based on the similarity between the benchmark datasets and the new dataset.

To make this idea more precise, we cast 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 1.1 for an illustration.

This work is an updated version of a previously published work under the same name [5] with additional baselines. My role in this joint work is the implementation and analysis of all experiments with the exception of 4.4.

Chapter 2

Related work

2.1 Benchmarking

Comparing models or algorithms across various tasks is a standard practice in ML and AI literature. Prior to Foundation Models [6], 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 [7] 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 [8]–[10]. In the context of Large Language Models, many benchmarks [3], [4], [11]–[15] 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.

2.2 Model selection

Selecting the best model, or model selection, is a classical topic in statistics and ML [16]–[18]. 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 [2], [19]. 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.

2.3 Out-of-distribution model selection

Recognizing the limitations of the model selection methods for in-distribution test data [20], [21], 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 [22], reweighing [23], [24], agreement of models or ensembles [25]–[27], or aligning model accuracy in-distribution with a confidence threshold [28]–[30]. 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.

2.4 Routing LLMs

Prior work on selecting LLMs primarily considers choosing one that produces the best generation for a given input. [15], [31], [32] 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 [33] 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. 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.

Chapter 3

Model Routing Theory

3.1 Binary Classifier Formulation

We start by introducing notation to describe the majority of NLP benchmarks. Let

$$\{x_1^d, \ldots, x_{n_d}^d\}_{d=1}^D$$

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, $m \in \{1, \ldots, M\}$, on the benchmark, for each task $d = 1, \ldots, D$, 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 [1]

$$\{f_{im}^d = F_d(x_i^d, o_{im}^d, r_i^d)\}_{i=1}^{n_d}$$

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.

3.1.1 Supervised learning from benchmarks

Our goal is to learn a simple routing function $g_m(x)$ for each LLM, $m = 1, \ldots, M$, 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

[1] 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.

them to choose the best model [15], [31], [32]. 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 [34]–[36], the notion of correctness can help to account for the heterogeneity of popular metrics and task difficulty levels. In Section 4.4, we also present results with raw metrics instead of correctness.

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

$$\min_{g_m} \sum_{d=1}^{D} \sum_{i=1}^{n_d} \ell(g_m(x_i^d), y(x_i^d, m)), \tag{3.1}$$

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 $P(y(x, m) = 1|x)$.

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, [37], [38] proposed methods for improving OOD generalization when training on data from multiple domains, while [21] 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 [39] and use a k-nearest neighbors classifier [40] 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?*

3.2 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:

$$\arg\max_m \tilde{S}(m, d'), \text{ where } \tilde{S}(m, d') = \frac{1}{n^{d'}} \sum_{i=1}^{n_{d'}} y(x_i^{d'}, m). \tag{3.2}$$

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

$$S_1(m, d') = \frac{1}{n^{d'}} \sum_{i=1}^{n_{d'}} g_m(x_i^{d'}), \tag{3.3}$$

but prior work has shown that accurately estimating $P(y|x)$, i.e., calibration, is challenging on OOD data [41]. 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 [42]. This motivates another score:

$$S_2(m, d') = \frac{1}{n^{d'}} \sum_{i=1}^{n_{d'}} \bar{g}_m(x_i^{d'}), \text{ where } \bar{g}_m(x_i^{d'}) = \mathbb{I}(g_m(x_i^{d'}) > t), \tag{3.4}$$

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., lower accuracy on OOD data from task d'. To address this issue, we model the out-of-distribution confidence of the predictions $\bar{g}_m$.

3.2.1 A simple OOD confidence model

We model LLM correctness as follows:

$$y(x, m)|x, d' = \begin{cases} \bar{g}_m(x) & \text{with probability } p(d', m) \\ 1 - \bar{g}_m(x) & \text{with probability } 1 - p(d', m), \end{cases} \tag{3.5}$$

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:

$$y(x, m)|x, d' \sim \text{Bern}(\bar{g}_m(x)p(d', m) + (1 - \bar{g}_m(x))(1 - p(d', m))). \tag{3.6}$$

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, [29] learn a threshold on model confidence, which is hard to apply when using kNN classifiers, and [27] 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 the following section.

Finally, given the model of LLM correctness 3.6, $\tilde{\mathbf{S}}(m, d')$ 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:

$$S_3(m, d') = S_2(m, d')p(d', m) + (1 - S_2(m, d'))(1 - p(d', m)). \tag{3.7}$$

When selecting an LLM with S_3, we consider an alternative to the arg max criterion based on our correctness model 3.6, which defaults to the best model on average across benchmark datasets when we are not sufficiently confident that a candidate model will be better:

$$\begin{cases} m_3 & \text{if } P(\tilde{\mathbf{S}}(m_3, d') > \tilde{\mathbf{S}}(m^*, d')) > \eta \\ m^* & \text{otherwise,} \end{cases} \tag{3.8}$$

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. The 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. Instead, we estimate the value of the expression via Monte Carlo sampling from the corresponding Bernoulli distributions. In all past and future experiments, we set $\eta = 0.6$.

3.2.2 Correctness predictors

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 [39]. For all completed experiments we have used `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:

$$g_m\left(x_i^{d'}\right) = \frac{1}{k} \sum_{e \in \mathrm{NN}(\phi(x_i^{d'}), k, \mathcal{D})} y(e, m), \tag{3.9}$$

where $y(e, m) \in \{0, 1\}$ is the correctness of model m on the (embedded) input e, and $\mathrm{NN}(\phi(x_i^{d'}), k, \mathcal{D})$ 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 3.4, 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 3.3 and equation 3.4, 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 3.7. This method comprises a dataset distance and a kernel smoother, defined as follows.

3.2.3 Dataset distance

Our dataset distance $u(d)$ is a one-sided variant of the Chamfer distance with extended neighborhood size. We define it formally below:

$$u(d) = \frac{1}{n_d} \sum_{i=1}^{n_d} nn(x_i^d, \mathcal{D}_{-d}), \tag{3.10}$$

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 $\mathcal{D}_{-d}$:

$$nn(x, \mathcal{D}) = \frac{1}{\kappa} \sum_{e \in \text{NN}(\phi(x), \kappa, \mathcal{D})} \text{cosine}(\phi(x), e), \tag{3.11}$$

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, which is a value we found performed well in early experiments.

3.2.4 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 $\mathcal{D}_{-d}$. 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:

$$p(d', m) = \frac{\sum_{z=1}^{Z} p_z(m) \mathcal{K}(u(d'), u_z)}{\sum_{z=1}^{Z} \mathcal{K}(u(d'), u_z)}, \tag{3.12}$$

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 a value we found to perform well through early 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.

3.3 Mixture of experts

Prior work has shown pooling the outputs of multiple models on every instance can greatly improve performance [43]. Loosely inspired by these ideas, we include an additional strategy we call Mixture of Experts (MoE). Instead of determining a score and selecting a model for a dataset, we route on a per instance level. As such, for an instance x_i we select the model $m_i = \arg\max_m g_m(x_i))$. For a new task d', the accuracy of this strategy would be:

$$\tilde{\text{MoE}}(d') = \frac{1}{n^{d'}} \sum_{i=1}^{n_{d'}} y(x_i^{d'}, \arg\max_m g_m(x_i^{d'})).$$

If $g_m(x_i^{d'})$ is tied between multiple models, we select the tied model that performs best on the training data.

Chapter 4

Model Routing Experiments

4.1 Model Routing on HELM

We select 29 datasets from the HELM benchmark [3] representing scenarios such as question answering (including a subset of MMLU [4]), text classification, language, knowledge, and reasoning, among others. A full list of HELM benchmarks and candidate LLMs can be found in 4.1 and 4.4 respectively.

4.1.1 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 3.2 to the performance of `llama-2-70b`, i.e., the BMA. All correctness predictors g_ms are kNN classifiers with $k = 5$.

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.

4.1.2 Main 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 4.1 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

Table 4.1: 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. 3.3	0.662	0.855	0.685	0.465	0.17	40.3B	6.172
S_2 eq. 3.4	0.676	0.868	0.636	0.468	0.10	44.3B	5.897
S_3 eq. 3.7, 3.8	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

(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. MoE chooses small models, but has low performance compared to other scores. 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.

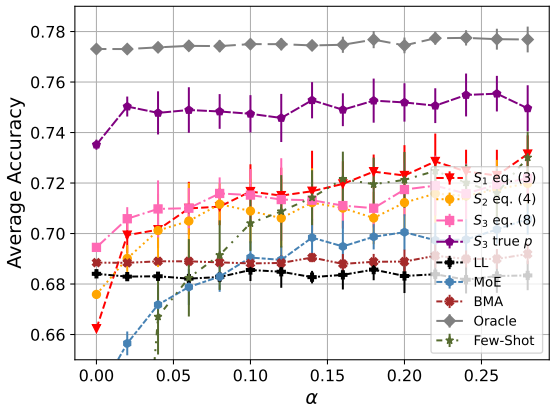


Figure 4.1: OOD Experiment: Using $\min(\alpha n^{d'}, 50)$ training samples from d' to reduce OOD gap.

4.1.3 Reducing the OOD gap

The average accuracy of correctness predictors across tasks and models for the experiments in Table 4.1 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.

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. 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 4.1 ($\alpha = 0$ corresponds to Table 4.1). Furthermore, we add an additional baseline, Few-shot, which selects the best performing model on the samples. This baseline performs considerably poorer for small values of α and approach the performance of S_1 for high values of α. Results for Pearson correlation are in Figure 4.3(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 4.3(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 $P(y|x)$, which are used by S_1 in equation 3.3. 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 [44] can help to improve LLM routing further.

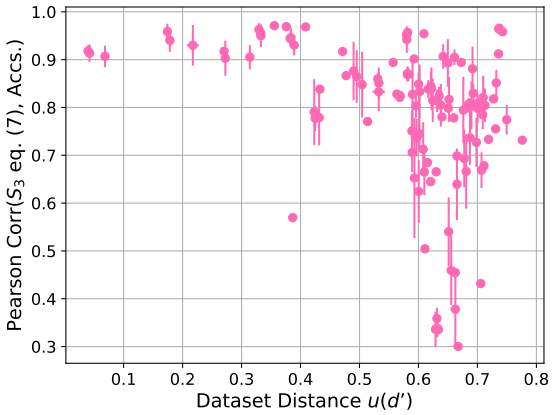


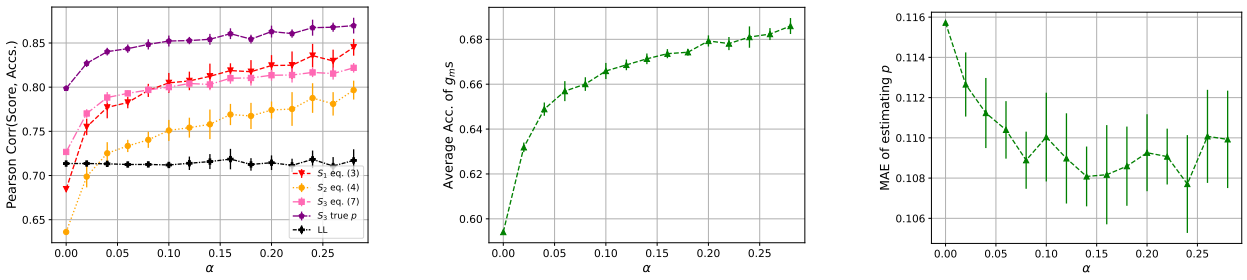
Figure 4.2: OOD Experiment: Correlation(S_3, Accs.) and $u(d')$.

4.2 Scores and accuracy correlation

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 4.2 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 4.1. For smaller distance values the correlation is approaching 1, while for large distances it sometimes deteriorates.

4.2.1 Additional results

We present additional results for this experiment in Figure 4.3. (a) shows Pearson correlation improvement as we increase α, similar to the trends in accuracy improvement in Figure 4.1; (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.



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

Figure 4.3: OOD Experiment: Additional results for Reducing the OOD gap experiment in Figure 4.1.

4.3 The Efficacy of small LLMs

As discussed in 4.1, 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.

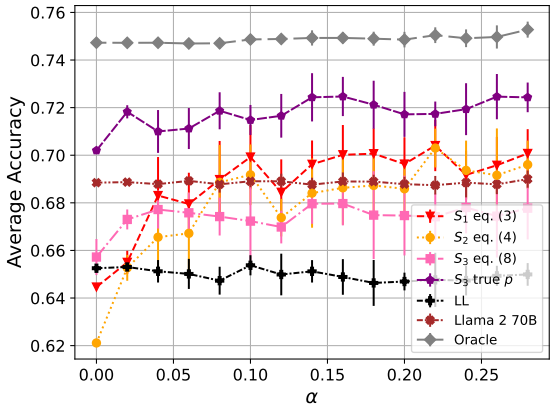


Figure 4.4: Small LLMs Experiment: LLM routing with $\leq$ 13B parameter models compared to `Llama 2 70B`.

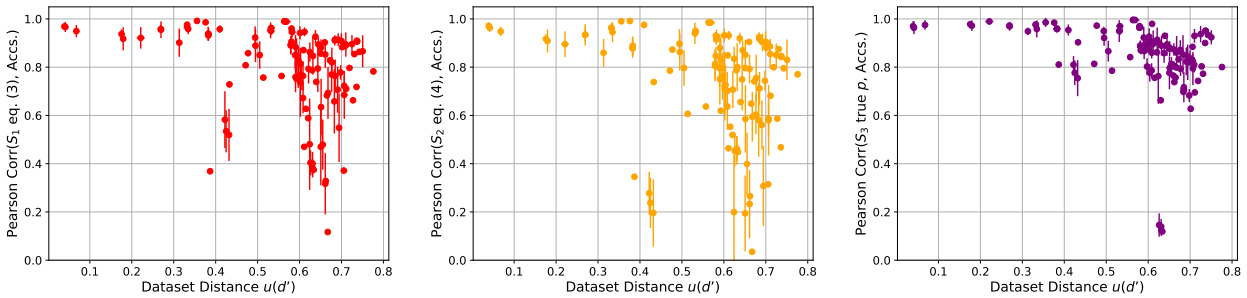
To evaluate the potential of smaller LLMs we revisit our HELM experiment in Figure 4.1. In Figure 4.4, we perform LLM routing using *only models with* $\leq$ *13B parameters* and compare it to the performance of `Llama 2 70B`. Oracle's performance demonstrates that it is conceptually possible to outperform a large model by routing smaller LLMs. Results with our scores S_1 and S_2 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.

This has interesting implications when considering the cost of LLM performance. We show that the relationship between LLMs performance and task is multifaceted. For a user, the path to improving LLM performance may not be to use a larger more expensive model, but rather to collect samples and use said samples to determine which smaller model is optimal.

4.3.1 Dataset distance and Pearson correlation

The dataset distance $u(d')$ is computed as in equation 3.10. As evident from equation 3.11, 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 4.5 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 4.2.



(a) S_1 equation 3.3 (b) S_2 equation 3.4 (c) S_3 equation 3.7

Figure 4.5: OOD Experiment: Correlation of scores and LLM accuracies on new tasks and corresponding data distances.

4.4 Model routing on Mix-Instruct

We further demonstrate our approach in a different setting and task type, on the MixInstruct benchmark dataset [15]. 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 [34], BARTScore [36], and BLEURT [35]. In [15], 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. Due to the per-instance setting, and since the test set was constructed from in-distribution data, we focus on our simplest router model S_1, equation 3.3.

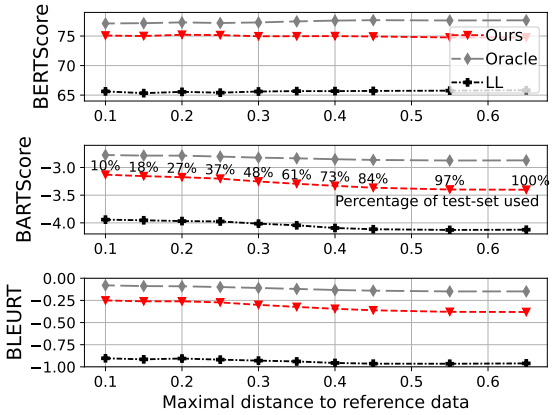


Figure 4.6: MixInstruct Experiment: Average metrics on subsets of the MixInstruct test set, defined by limiting the maximal average distance between test instances and their closest neighbors in the reference (train) set.

We compare our approach with the scoring methods examined by [15], as well as scoring based on the average log-likelihood (LL) of the model responses to the inputs. Additionally, we present the metrics for the best models on average (BMA), Open-Assistant [45] and Vicuna [46]. We report the results of BERTScore, BARTScore and BLEURT in Table

Table 4.2: MixInstruct Experiment: 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 [47]	64.77	-4.03	-0.88	N
SimCLS [31]	73.14	<u>-3.22</u>	<u>-0.38</u>	N
SummaReranker [32]	71.60	-3.25	-0.41	N
PairRanker [15]	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

4.2, 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.

4.4.1 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 4.6 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.

Table 4.3: 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 4.4: 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	

Chapter 5

Conclusion

In this work we have explored the idea of using benchmark datasets to route models and investigated the problem of out-of-distribution datasets. Our results show that by using a simple method to select models, it is possible to outperform the best overall model. Additionally, we improved performance with a strategy to overcome the challenge of out-of-distribution datasets through use of dataset-distance. While this work covers the entirety of my involvement with these efforts, this avenue of research is far from concluded. My collaborators are exploring adding additional datasets to improve the performance of our method. During my time working on this thesis, I was involved in other work on the costs of LLMs which built upon the ideas presented in this work. I wish my collaborators the best of luck as they continue their work in this domain, and I am excited to learn of the results that they find.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, *et al.*, "Holistic evaluation of language models," *arXiv preprint arXiv:2211.09110*, 2022.

[4] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2020.

[5] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, and M. Yurochkin, *Large language model routing with benchmark datasets*, 2023. arXiv: 2309.15789 [cs.CL].

[6] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[7] M. Kelly, R. Longjohn, and K. Nottingham, *The UCI Machine Learning Repository*, 2023. [Online]. Available: https://archive.ics.uci.edu.

[8] O. Bojar, C. Buck, C. Federmann, B. Haddow, P. Koehn, J. Leveling, C. Monz, P. Pecina, M. Post, H. Saint-Amand, *et al.*, "Findings of the 2014 workshop on statistical machine translation," in *Proceedings of the ninth workshop on statistical machine translation*, 2014, pp. 12–58.

[9] P. Goyal, D. Mahajan, A. Gupta, and I. Misra, "Scaling and benchmarking self-supervised visual representation learning," in *Proceedings of the ieee/cvf International Conference on computer vision*, 2019, pp. 6391–6400.

[10] C. Li, H. Liu, L. Li, P. Zhang, J. Aneja, J. Yang, P. Jin, H. Hu, Z. Liu, Y. J. Lee, *et al.*, "Elevater: A benchmark and toolkit for evaluating language-augmented visual models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9287–9301, 2022.

[11] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[12] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems*, vol. 32, 2019.

[13] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro, A. Gupta, A. Garriga-Alonso, *et al.*, "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models," *arXiv preprint arXiv:2206.04615*, 2022.

[14] E. Beeching, C. Fourrier, N. Habib, S. Han, N. Lambert, N. Rajani, O. Sanseviero, L. Tunstall, and T. Wolf, *Open llm leaderboard*, https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard, 2023.

[15] D. Jiang, X. Ren, and B. Y. 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)*, Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 14 165–14 178. DOI: 10.18653/v1/2023.acl-long.792. [Online]. Available: https://aclanthology.org/2023.acl-long.792.

[16] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4.

[17] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[18] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.

[19] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[20] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," in *International Conference on Learning Representations*, 2021.

[21] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5637–5664.

[22] H. Xu and R. Tibshirani, "Estimation of prediction error with known covariate shift," *arXiv preprint arXiv:2205.01849*, 2022.

[23] M. Chen, K. Goel, N. S. Sohoni, F. Poms, K. Fatahalian, and C. Ré, "Mandoline: Model evaluation under distribution shift," in *International conference on machine learning*, PMLR, 2021, pp. 1617–1629.

[24] S. Maity, M. Yurochkin, M. Banerjee, and Y. Sun, "Understanding new tasks through the lens of training data via exponential tilting," in *International Conference on Learning Representations*, 2023.

[25] Y. Jiang, V. Nagarajan, C. Baek, and J. Z. Kolter, "Assessing Generalization of SGD via Disagreement," in *International Conference on Learning Representations*, 2021.

[26] J. Chen, F. Liu, B. Avci, X. Wu, Y. Liang, and S. Jha, "Detecting errors and estimating accuracy on unlabeled data with self-training ensembles," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 980–14 992, 2021.

[27] N. H. Ng, N. Hulkund, K. Cho, and M. Ghassemi, "Predicting out-of-domain generalization with neighborhood invariance," *Transactions on Machine Learning Research*, 2023.

[28] D. Guillory, V. Shankar, S. Ebrahimi, T. Darrell, and L. Schmidt, "Predicting with confidence on unseen distributions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 1134–1144.

[29] S. Garg, S. Balakrishnan, Z. C. Lipton, B. Neyshabur, and H. Sedghi, "Leveraging unlabeled data to predict out-of-distribution performance," in *International Conference on Learning Representations*, 2022.

[30] Y. Yu, Z. Yang, A. Wei, Y. Ma, and J. Steinhardt, "Predicting out-of-distribution error with the projection norm," in *International Conference on Machine Learning*, PMLR, 2022, pp. 25 721–25 746.

[31] Y. Liu and P. 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)*, 2021, pp. 1065–1072.

[32] M. Ravaut, S. Joty, and N. 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)*, 2022, pp. 4504–4524.

[33] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance," *arXiv preprint arXiv:2305.05176*, 2023.

[34] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating Text Generation with BERT," in *International Conference on Learning Representations*, 2020.

[35] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7881–7892.

[36] W. Yuan, G. Neubig, and P. Liu, "Bartscore: Evaluating generated text as text generation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 263–27 277, 2021.

[37] B. Sun and K. 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*, Springer, 2016, pp. 443–450.

[38] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[39] N. Reimers and I. 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.

[40] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[41] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Advances in neural information processing systems*, vol. 32, 2019.

[42] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*, PMLR, 2017, pp. 1321–1330.

[43] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, *Towards understanding mixture of experts in deep learning*, 2022. arXiv: 2208.02813 [cs.LG].

[44] B. Settles, "Active learning literature survey," 2009.

[45] LAION-AI, *Open assistant*, 2023. [Online]. Available: https://github.com/LAION-AI/Open-Assistant.

[46] W.-L. Chiang, Z. Li, Z. Lin, *et al.*, *Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality*, Mar. 2023. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/.

[47] J. Salazar, D. Liang, T. Q. Nguyen, and K. Kirchhoff, "Masked language model scoring," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 2699–2712.