Research Article

SMOOTHIE: Label Free Language Model Routing

Neel Guha[1], Mayee F. Chen[1], Trevor Chow[1], Ishan S. Khare[1], Christopher Ré[1]

1. Department of Computer Science, Stanford University, United States

Large language models (LLMs) are increasingly used in applications where LLM inputs may span many different tasks. Recent work has found that the choice of LLM is consequential, and different LLMs may be good for different input samples. Prior approaches have thus explored how engineers might select an LLM to use for each sample (i.e. *routing*). While existing routing methods mostly require training auxiliary models on human-annotated data, our work explores whether it is possible to perform *unsupervised* routing. We propose SMOOTHIE, a weak supervision-inspired routing approach that requires no labeled data. Given a set of outputs from different LLMs, SMOOTHIE constructs a latent variable graphical model over embedding representations of observable LLM outputs and unknown "true" outputs. Using this graphical model, we estimate sample-dependent quality scores for each LLM, and route each sample to the LLM with the highest corresponding score. We find that SMOOTHIE's LLM quality-scores correlate with ground-truth model quality (correctly identifying the optimal model on 9/14 tasks), and that SMOOTHIE outperforms baselines for routing by up to 10 points accuracy.

Neel Guha and Mayee F. Chen equally contributed to this work.

Corresponding authors: Neel Guha, nguha@stanford.edu; Mayee F. Chen, mfchen@stanford.edu

1. Introduction

Large language models (LLMs) are increasingly being deployed in *multi-capability* regimes where data inputs may span a diverse range of tasks, each of which requires different capabilities[1]. For instance, an LLM-powered chatbot may be asked to write code, answer questions about different domains, summarize documents, perform extraction, and more[2][3][4][1]. One challenge is that while engineers often have access to numerous pre-trained LLMs (i.e., through Huggingface or various APIs), they do not know which LLM is optimal for each possible user input[5]. Because the quality of generations can vary significantly between LLMs, choosing the right LLM for each input sample is important to ensure high task performance[6].

Recent work has explored various ways to utilize ensembles of pretrained LLMs in multi-capability settings, by (1) collecting a diverse pool of LLMs and (2) identifying which LLM to *route* each sample to[5][7]. However, most

existing approaches require labeled data; Engineers typically either (1) train an auxiliary model using labeled data to rank or predict the LLM to which each sample should be routed[6][8], or (2) directly use labeled data to determine which LLM is the best on average[5]. As a result, engineers designing routing protocols face the practical difficulty of constructing labeled datasets.

Given a candidate pool of LLMs and an unlabeled test dataset, this paper explores how to best select LLM outputs for each sample in an entirely unsupervised manner—without labeled data, or models trained on labeled data. To make progress in addressing this question, we face two technical challenges:

- **Unknown LLM quality:** The first challenge is estimating the quality of each LLM. Access to labeled data allows engineers to identify higher performing LLMs by measuring the accuracy/quality of LLM outputs. In this paper, we study the question of how to estimate quality *without* labeled validation data.
- **Sample-conditional generator performance:** The second challenge is determining how to select the best LLM for each individual test sample. LLM outputs can vary in quality over different samples, which could render population-level estimates of LLM quality misleading.

In this work, we propose SMOOTHIE, a method for routing samples to LLMs in a label-free manner (Figure 1). Below, we describe how SMOOTHIE addresses the two challenges described above.

- **Quality estimation:** Using the LLM outputs for each test sample as "voters," SMOOTHIE estimates the quality of each generator using methods from Weak Supervision (WS). Concretely, SMOOTHIE constructs a latent variable graphical model over observable LLM outputs and an unknown true output. By modeling the embedding vector difference between each LLM output and the true output as a multivariate Gaussian, we can derive a closed-form estimator adapted from[9] for learning LLM quality scores efficiently.
- **Conditioning:** We condition these quality estimates to be particular to a given test sample by only using the nearest neighbors of a test sample in the training data as inputs to the estimator (i.e., kernel smoothing). We then route each test sample to the LLM with the highest quality score estimate on that sample. We call the version of SMOOTHIE that produces quality estimates using all available test data SMOOTHIE-GLOBAL, and we call the version that uses a sample's nearest neighbors SMOOTHIE-LOCAL.

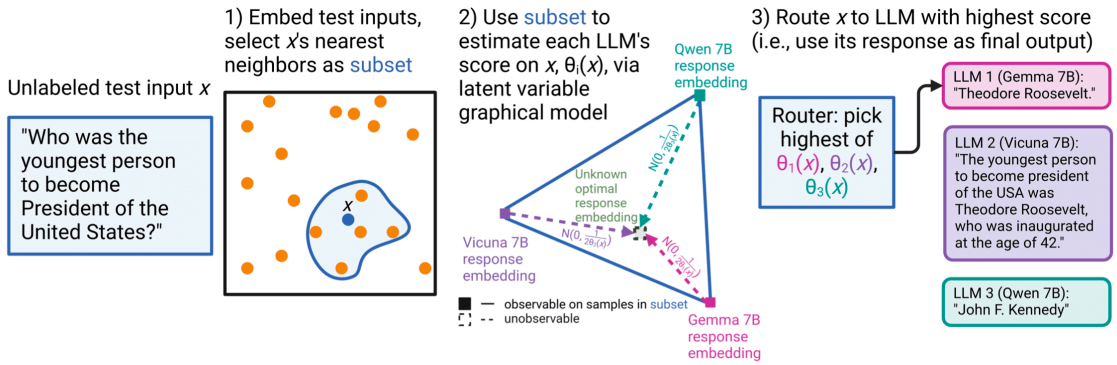

Figure 1. For a given input x, SMOOTHIE estimates the quality of every LLM ensemble's generation, and uses this quality weight to route x to a single LLM.

We empirically evaluate SMOOTHIE in three stages.

- **LLM selection:** First, we assess SMOOTHIE-GLOBAL's ability to identify—from an ensemble of mixed quality LLMs—the optimal LLM for a given task overall. On traditional generation tasks such as summarization, reading comprehension, and data-to-text generation, we find that SMOOTHIE-GLOBAL's learned LLM quality-weights correlate with actual LLM performance ($\rho = 0.72$)), and on the AlpacaEval benchmark, SMOOTHIE-GLOBAL identifies the best-performing instruction model 70% of the time[10]. The highest quality LLM identified by SMOOTHIE-GLOBAL—all computed without labeled data—can beat random-selection by up to 15 points win-rate on AlpacaEval, and by up to 8 points on SQuAD.
- **Routing:** Second, we study whether SMOOTHIE-LOCAL's sample-conditional scoring mechanism allows it to *route*samples in mixed-task datasets to higher-performing LLMs (i.e., the multi-capability regime). We find that SMOOTHIE-LOCAL can improve the quality of produced generations by up to 7 points accuracy over SMOOTHIE-GLOBAL, and that SMOOTHIE-LOCAL outperforms baseline unsupervised routing methods by up to 10 points accuracy and supervised routing methods by up to 5.0 points accuracy.
- **Prompt selection:** Finally, we assess whether SMOOTHIE's quality-estimation mechanism can be applied to select the optimal prompt template in a candidate pool while using a fixed LLM. We find that SMOOTHIE-GLOBAL can outperform other prompt selection approaches by up to 18 points, allowing a 410M parameter model to match the performance of 6.9B parameter model.

2. Related Work

We provide an abbreviated related work, with a full treatment in Appendix C.

Routing

Routing has been classically utilized in Mixture-of-Experts models[11][12][13][14], which involve jointly training a set of models as well as a router. Recently, routing mechanisms have been used at inference time to decide which pre-trained LLM to use for a given sample[8]. Some approaches involve training an auxiliary model using labeled training data to either score or rank the performance of each LLM on each test sample[15][16]. Others do not involve training a model but instead use nearest neighbor methods, selecting the LLM that does the best on a test sample's labeled neighbors[17][5]. In contrast, SMOOTHIE does not require any labels.

Ensembling

Ensembling is another way of utilizing a pool of LLMs. Existing work has primarily focused on ensembling outputs for classification tasks[18][19][20]. Ensembling generative outputs typically requires training an auxiliary model[6], combining or switching among outputs when decoding[21][22], or averaging in weight space[23].

Prompt selection

In addition to selecting the best LLM for a sample, prior works have studied how to select the best prompt or in-context examples. While the simplest approach is to use a held-out labeled dataset[24], there are also retrieval-based approaches to selecting the best in-context examples[25], as well as approaches based on mutual information[26] and probability-based measures[27], although the latter two are limited to classification.

Weak supervision

SMOOTHIE utilizes statistical techniques inspired by weak supervision, which programmatically generate labels for an unlabeled dataset by aggregating the predictions of several weak "voters" via a latent variable graphical model[28][29]. Weak supervision has mostly been studied in classification settings[30][31] but more recently has been extended to tasks such as learning rankings and manifolds[9][32]. We derive our estimation procedure from the Gaussian model in[9], applying it to LLM embeddings and the routing setting.

3. Preliminaries

3.1. Problem setup

Let $\mathcal{V}$ be the token vocabulary space, and let $\overline{\mathcal{V}} = \mathcal{V} \times \cdots \times \mathcal{V}$ be the space of all vocabulary sequences. We consider a generative task with input text $x \in \mathcal{X} \subset \overline{\mathcal{V}}$ and reference output text $y \in \mathcal{Y} \subset \overline{\mathcal{V}}$. We have a candidate pool of m LLMs, $G = \{g_1, \ldots, g_m\}$, where each $g_i \in \mathcal{G} : \mathcal{X} \to \mathcal{Y}$ produces a generative output sequence $g_i(x)$ for a given input text sequence x. We are given an unlabeled test dataset $\mathcal{D}_{\text{test}} = \{x_i\}_{i=1}^n$, where the ground-truth reference outputs are *unknown*.

Our goal is to route each sample $x \in \mathcal{D}_{\text{test}}$ to one of the LLMs in G. Specifically, we wish to construct a router $\text{route} : \mathcal{G}^m \times \mathcal{X} \to \mathcal{G}$ that selects the LLM that yields the highest quality generation on x for each test sample x, without any labeled data.

3.2. Graphical model

We present a probabilistic graphical model (see Figure 1 (center)) that describes how the LLM outputs, $g_1(x), \ldots, g_m(x)$, are related to a true output y in terms of each LLM's quality on a given input x, which we call $\theta_i(x)$, corresponding to each $g_i(x)$. Let $z_{g_0} : \overline{\mathcal{V}} \to \mathbb{R}^d$ map from a sequence of tokens to a d-dimensional embedding using a common model g_0 such as SentenceBERT[33]. Define $\lambda_i(x) := z_{g_0}([x, g_i(x)])$ to be the observable embedding of x and the LLM output, and define $z^\star(x) := z_{g_0}([x, y])$ to be the latent ground-truth embedding of x and reference output y. Similar to the approach in[9], we model the distribution over embedding vectors, $\Pr(z^\star(x), \lambda_1(x), \ldots, \lambda_m(x)|x)$ as

$$\Pr(z^\star(x), \lambda_1(x), \ldots, \lambda_m(x)|x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{m} -\theta_i(x) \|\lambda_i(x) - z^\star(x)\|^2 \right) \tag{1}$$

where Z is the log partition function and the $\theta_i(x)$s—the LLM quality scores—are canonical parameters of the graphical model. Intuitively, our model captures LLM quality by supposing that if g_i is of high quality and $\theta_i(x)$ is very large, then it should be unlikely for the LLM output to be very different from the true output in terms of Euclidean distance in embedding space. Conversely, if $\theta_i(x)$ is small, we assign larger probability to the setting where $\lambda_i(x)$ and $z^\star(x)$ differ significantly. Finally, note that this graphical model corresponds to a multivariate Gaussian. That is, the vector $[\lambda_1(x) - z^\star(x), \ldots, \lambda_m(x) - z^\star(x)] \in \mathbb{R}^{dm}$ is Gaussian with mean $\mu = \vec{0}$ and a diagonal covariance matrix $\Sigma \in \mathbb{R}^{dm \times dm}$ with $\Sigma_{jj} = \frac{1}{2\theta_{\lceil j/m \rceil}(x)}$. Intuitively, this means that the average difference vector between each λ_i and $z^\star(x)$ is centered, with its magnitude inversely proportional to the LLM score $\theta_i(x)$ and independent of other LLMs. Given this probabilistic graphical model, our goal is to learn each quality score $\theta_i(x)$ from the unlabeled test dataset and use these for improved routing.

4. Method

Given an unlabeled test dataset $\mathcal{D}_{\text{test}}$ and a pool of LLMs G, SMOOTHIE consists of two steps:

1. **Estimation:** The LLM quality scores $\theta_1(x), \ldots, \theta_m(x)$ are learned for each $x \in \mathcal{D}_{\text{test}}$ (Section 4.1, Algorithm 1).
2. **Routing:** The LLM with the highest scores is selected, and its output is used as our final prediction for x (Section 4.2).

We describe each step in the following sections.

Algorithm 1 ESTIMATE SCORES

Input: unlabeled test dataset $\mathcal{D}_{\text{test}}$, LLMs G, n_0 nearest neighbors parameter, g_0 embedding model with dimension d.
For all $x \in \mathcal{D}_{\text{test}}$ and $g_i \in G$, obtain the generator output $\vec{g}_i(x)$ and embed the input and generator output using model g_0 to get embedding $\lambda_i(x) := z_{g_0}([x, g_i(x)])$.
for $x \in \mathcal{D}_{\text{test}}, g_i \in G$ **do**
    for $j, k \neq i \in [m]$ **do**
        Compute $\hat{\delta}_{ij}(x) = \frac{1}{n_0} \sum_{x' \in \text{NN}_{n_0}(x)} \|\lambda_i(x') - \lambda_j(x')\|^2$, and similarly $\hat{\delta}_{ik}(x)$ and $\hat{\delta}_{jk}(x)$.
        Set $\hat{\theta}_i^{jk}(x) = d/(\hat{\delta}_{ij}(x) + \hat{\delta}_{ik}(x) - \hat{\delta}_{jk}(x))$.
    Compute averaged estimate $\hat{\theta}_i(x) = \frac{1}{\binom{m-1}{2}} \sum_{j,k \neq i} \hat{\theta}_i^{jk}(x)$.
return $\hat{\theta}_i(x)$ for all $x \in \mathcal{D}_{\text{test}}, g_i \in G$.

Algorithm 1. Estimate Scores

4.1. LLM score estimation

We describe how to estimate each $\theta_i(x)$s in the graphical model in (1) using only unlabeled data from $\mathcal{D}_{\text{test}}$. Then, we describe how the LLM score estimate can be instantiated to be sample-conditional.

Computing $\theta_i(x)$

Below, we state a simple property arising from the fact that (1) corresponds to a multivariate Gaussian with a diagonal covariance matrix.

Proposition 1.[9] *For any $i, j \in [m]$, it follows from the graphical model in (1) that*

$$\mathbb{E}\|\lambda_i(x) - \lambda_j(x)\|^2 = \mathbb{E}\|\lambda_i(x) - z^\star(x)\|^2 + \mathbb{E}\|\lambda_j(x) - z^\star(x)\|^2. \tag{2}$$

The proof is in Appendix D and relies on the fact that off-diagonal entries of Σ are 0. Note that the left hand side of the equation is observable while the two expectations on the right are unknown. We can apply this equation to pairs of LLM embeddings over a triplet of $\lambda_i, \lambda_j, \lambda_k$ to form a system of three equations with three unknown expectations. Solving, we have

$$\mathbb{E}\left[\|\lambda_i(x) - z^\star(x)\|^2\right] = \frac{1}{2}(\delta_{ij}(x) + \delta_{ik}(x) - \delta_{jk}(x)) \quad \forall (i,j,k) \in [m], \tag{3}$$

where $\delta_{ij}(x) = \mathbb{E}\left[\|\lambda_i(x) - \lambda_j(x)\|^2\right]$. Since (1) is a multivariate Gaussian with $\Sigma_{jj} = \frac{1}{2\theta_{[j/m]}(x)}$, we can write $\theta_i(x)$ as the following function of $\mathbb{E}\left[\|\lambda_i(x) - z^\star(x)\|^2\right]$:

$$\mathbb{E}\left[\|\lambda_i(x) - z^\star(x)\|^2\right] = \sum_{j=1}^{d} \mathbb{E}\left[(\lambda_{i,j}(x) - z_j^\star(x))^2\right] = \sum_{j=1}^{d} \text{Var}(\lambda_{i,j}(x) - z_j^\star(x)) = \frac{d}{2\theta_i(x)}, \tag{4}$$

where $\lambda_{i,j}(x)$ and $z_j^\star(x)$ are the jth indices of the embeddings $\lambda_i(x)$ and $z^\star(x)$ respectively. Therefore, we can write $\theta_i^{jk}(x) = \frac{d}{\delta_{ij}(x) + \delta_{ik}(x) - \delta_{jk}(x)}$, where each $\delta_{ij}(x)$ can be estimated using the LLM outputs on $\mathcal{D}_{\text{test}}$, and in practice in Algorithm 4.1 we estimate $\theta_i(x)$ by averaging $\theta_i^{jk}(x)$ over all $\binom{m-1}{2}$ pairs of $(j,k) \neq i$.

Sample-conditional estimation of $\theta_i(x)$

Note that the expectation in $\delta_{ij}(x) = \mathbb{E}\left[\|\lambda_i(x) - \lambda_j(x)\|\right]$ is over the randomness in $\lambda_i(x), \lambda_j(x)$ conditioned on a fixed point x. However, we only have one sample per x. One simple approach is to use the entire dataset to estimate $\theta_i(x)$, i.e., $\hat{\delta}_{ij}(x) = \frac{1}{n}\sum_{x' \in \mathcal{D}_{\text{test}}} \|\lambda_i(x') - \lambda_j(x')\|^2$. We denote this as SMOOTHIE-GLOBAL. However, in SMOOTHIE-GLOBAL each $\theta_i(x)$ for $i \in [m]$ is a constant over the entire $\mathcal{D}_{\text{test}}$. Therefore, we use nearest neighbor kernel smoothing to estimate each $\delta_{ij}(x)$ in a sample-dependent manner, an approach we call SMOOTHIE-LOCAL. Concretely, for $x \in \mathcal{D}_{\text{test}}$, define $\text{NN}_{n_0}(x) \subset \mathcal{D}_{\text{test}}$ as the $n_0 < n$ nearest neighbors of x(excluding x itself) in f_0's embedding space. Then, we construct $\hat{\delta}_{ij}(x) = \frac{1}{n_0}\sum_{x' \in \text{NN}_{n_0}(x)} \|\lambda_i(x') - \lambda_j(x')\|^2$, and do the same for $\hat{\delta}_{ik}(x), \hat{\delta}_{jk}(x)$ to get a sample-conditional estimate of $\theta_i(x)$. The procedure for estimating $\theta_i(x)$ in SMOOTHIE-LOCAL is outlined in Algorithm 4.1.

4.2. Routing

Once we have estimates of $\hat{\theta}_i(x)$ for each of the m generators by using Algorithm 1, we can construct our $\text{route}()$ function. We define $\text{route}(\mathcal{G}, x) = g_i$ where $i = \arg\max\{\theta_1(x), \ldots, \theta_m(x)\}$, which selects the highest scoring LLM for input x based on $\hat{\theta}_i(x)$. We apply this on $\mathcal{D}_{\text{test}}$ to determine the best LLM for each input sample.

5. Results

We empirically analyze SMOOTHIE-GLOBAL and SMOOTHIE-LOCAL, focusing on four questions:

1. How well does SMOOTHIE-GLOBAL recover ground-truth LLM rankings over samples belonging to the same task (Section 5.1)?
2. In multi-task datasets, how well can SMOOTHIE-LOCAL perform unsupervised-routing, by identifying the best LLM for each sample (Section 5.2)?
3. Can SMOOTHIE-GLOBAL and SMOOTHIE-LOCAL be applied to select from or route between different prompts (Section 5.3)?
4. How does SMOOTHIE-GLOBAL and SMOOTHIE-LOCAL's performance change as a function of different algorithmic choices (Section 5.4)?

5.1. Single-Task LLM Scoring

Setup

We begin by evaluating whether SMOOTHIE-GLOBAL can accurately learn the relative performance of different LLMs on a single task-dataset. We study three categories of tasks. First, we consider 7 datasets corresponding to commonly-studied natural language generation (NLG) tasks[34]: CNN/DailyMail and XSum (summarization), SQuAD (reading comprehension), TriviaQA (factual recall), E2E and WebNLG (data-to-text generation), and

LegalBench's Definition Extraction (text extraction)[35][36][37][38][39][40][41][42][43][4]. We report Rouge2 for summarization and data-to-text generation tasks and accuracy for all others. For all tasks other than Definition Extraction we evaluate SMOOTHIE-GLOBAL on a 1000 sample subset.[1] For these tasks, we consider two ensembles of LLMs at different size points. At the 3B size point, our ensemble consists of Pythia-2.8B[44], Gemma-2B[45], Incite-3B[46], and Dolly-3B[47]. At the 7B size point, our ensemble consists of Llama-2[48], Mistral[49], Vicuna[50], Gemma-7B[45], and Nous Capybara[51]. We manually write a single prompt template for each task, and all model generations rely on this template.

Second, we consider two instruction-following benchmarks: AlpacaEval and MixInstruct[10][52][53][6]. For AlpacaEval, we rely on responses accessible via the online leaderboard.[2] We identify 10 LLMs (each from a different base family), and download these models' responses to the AlpacaEval instructions. We conduct 10 different simulations, where in each simulation we randomly select 5 LLMs from our pool to function as an ensemble. Reported win-rates use the standard GPT-4 references. For MixInstruct, we use generations from an ensemble of 11 different LLMs originally studied in[6]. Following[6], we measure generation quality using a ChatGPT-based rank.

Finally, we consider a more "reasoning-intensive" task, GSM8K[54]. We consider an ensemble of three models: Gemma-7B, Phi-2[55], and Llema-7b[56]. We prompt each model to provide a chain-of-though reasoning[57], and apply SMOOTHIE to these generations.

For all datasets, we apply SMOOTHIE-GLOBAL using SentenceBERT (`all-mpnet-base-v2`) embeddings of generations[33].

Results

We first measure how frequently the highest-weighted LLM according to SMOOTHIE-GLOBAL corresponds to the best-performing LLM in the ensemble. We observe that SMOOTHIE-GLOBAL selects the best-performing LLM for 4/7 tasks on the 3B ensemble, and for 5/7 tasks on the 7B ensemble (Figure 10). On AlpacaEval, SMOOTHIE-GLOBAL selects the best-performing LLM by win-rate for 8/10 ensembles, and the best performing LLM by length-controlled win-rate for 7/10 ensembles. On MixInstruct and GSM8K, SMOOTHIE-GLOBAL again identifies the best-performing LLM in the ensemble.

Second, we measure how well SMOOTHIE-GLOBAL captures quality differences between LLMs in the ensemble, by computing the Spearman's rank correlation coefficient between θ_i and ground truth quality scores ensemble models. Overall, we find that SMOOTHIE-GLOBAL's learned weights approximate the relative ordering of model quality well. On the NLG tasks SMOOTHIE-GLOBAL we measure an average correlation coefficient (across both ensembles and seven tasks) of 0.72. Figure 2(a) visually depicts the distribution of task coefficients—on only one ensemble/dataset pair is there a correlation coefficient ≤ 0. On MixInstruct, we observe a correlation coefficient of 0.94, and on AlpacaEval, we observe a correlation coefficient of 0.46.

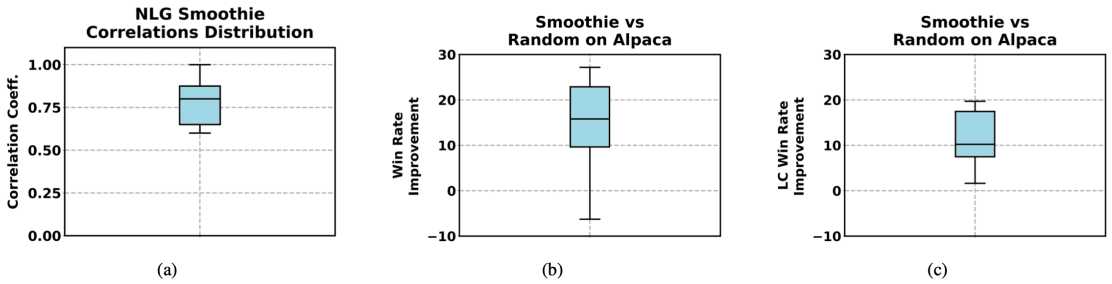

Figure 2. (a) Spearman's rank correlation coefficient between SMOOTHIE-GLOBAL weights and ground-truth LLM performance for 3B and 7B ensembles across NLG tasks. **(b)** SMOOTHIE-GLOBAL's improvement over RANDOM by win-rate on AlpacaEval. **(c)** SMOOTHIE-GLOBAL's improvement over RANDOM by length-controlled win-rate on AlpacaEval.

Finally, we measure how the performance of the LLM selected by SMOOTHIE compares to other selection algorithms. We first compare SMOOTHIE-GLOBAL to an unsupervised random baseline (Random), which would select a random model from the ensemble. We reported the expected performance of this method, which is equivalent to taking the average performance of the ensemble. We also compare SMOOTHIE-GLOBAL to a labeled baseline which simulates selecting an LLM on the basis of a small amount of validation data[24] (BEST-ON-VAL). We sample a small labeled validation set (50 samples) and select the LLM that performs the best on this set. To account for sampling variation, we repeat this with 10 random draws and report the average performance. Because AlpacaEval has no training split and MixInstruct has no labeled data, we only compare SMOOTHIE-GLOBAL to Random on those datasets.

Table 1 provides results for the seven NLG tasks. We find that SMOOTHIE-GLOBAL outperforms the unsupervised Randombaseline on 6/7 tasks for the 3B ensemble and on 7/7 tasks for the 7B ensemble. SMOOTHIE-GLOBAL outperforms Randomby up to 7pts (on tasks measured by rouge2), and by up to 12pts (on tasks measured by accuracy). We also observe that SMOOTHIE-GLOBAL is frequently competitive with and even outperforms the BEST-ON-VAL baseline, which uses labeled data. SMOOTHIE-GLOBAL outperforms BEST-ON-VAL on 4/7 tasks for the 3B ensemble, and 5/7 tasks for the 7B ensemble. On GSM8K, SMOOTHIE-GLOBAL achieves a solve-rate of 37.5% (matching BEST-ON-VAL, while Random achieves a solve-rate of 28.3% (Table 11).

		CNN	Def. Ext.	E2E	SQuAD	TriviaQA	WebNLG	XSum
3B	RANDOM	12.9	52.4	27.3	59.6	32.7	23.4	4.5
	SMOOTHIE-GLOBAL	**14.3**	**61.5**	**31.8**	60.7	32.1	**30.7**	4.5
	BEST-ON-VAL	13.0	60.5	31.1	**66.4**	**38.7**	30.3	**5.3**
7B	RANDOM	13.7	58.5	35.3	67.9	59.3	44.1	6.9
	SMOOTHIE-GLOBAL	**14.5**	**70.9**	**36.9**	**76.2**	**68.3**	45.9	**8.4**
	BEST-ON-VAL	**14.5**	69.4	36.7	74.0	65.8	**48.3**	8.3

Table 1. Comparing SMOOTHIE-GLOBAL to baseline methods on different ensembles across NLG datasets. Underlined values are the best performing *unsupervised* methods. Bold values are the best performing *overall* methods. We report rouge2 scores for CNN, XSum, WebNLG, and E2E, and accuracy for the rest. All metrics are scaled to 0-100.

SMOOTHIE-GLOBAL also outperforms the Random baseline on the instruction-following datasets. On MixInstruct, SMOOTHIE-GLOBAL achieves a ChatGPT-rank (↓) of 3.91, while Random achieves a ChatGPT-rank of 5.95 (Table 10). On AlpacaEval, SMOOTHIE-GLOBAL outperforms Random on all but one trial (across both win-rate and length-controlled win-rate). SMOOTHIE-GLOBAL outperforms Random by an average of 15pt win-rate, and up to 27pts. Figure 2(b) and Figure 2(c)e this distribution.

5.2. Multi-task Routing

Setup

We next assess whether SMOOTHIE-LOCAL's sample-conditional scoring mechanism allows it to route samples to LLMs in the multi-capability regime. We construct two mixed-task distributions by combining existing datasets. The first distribution corresponds to tasks measured by accuracy, and contains SQuAD, TriviaQA, and Definition Extraction. We refer to this as DISTR-ACC. The second distribution corresponds to tasks measured by Rouge2, and contains CNN/DailyMail, XSum, Web NLG, and E2E. We refer to this as DISTR-ROUGE2. For each mixed-task dataset, we report the metric averaged across all tasks. We compare to three baselines.

- RANDOM: A random-selection baseline which returns a generation from a random LLM in the ensemble. Though naive, prior work has found this to be a strong method in practice[58]. We run 10 trials and report the mean of this approach to account for variance.

- LABELED-KNN: A labeled data-based KNN baseline. For this, we sample 50 labeled samples from a separate hold-out set ($\mathcal{D}_{\text{val}}$), and measure the performance of each candidate LLM on this set. For a given test sample x, we identify the 20 most semantically similar instances in $\mathcal{D}_{\text{val}}$(using SentenceBERT embeddings[33]), and route x to the highest performing LLM over this subset. We note that the LABELED-KNN baseline is derived from routing methods in[5][17].
- PAIRRM: A reward model from[6] which accepts an instruction and multiple generations as input, scores each generations suitability for the instruction, and returns the predicted best generation. PAIRRM is a labeled-data method which[6] trained on collected preference data.

In addition, we also compare the best individual model in the ensemble (BEST-MODEL), and SMOOTHIE-GLOBAL. For both mixed-task datasets, we run SMOOTHIE-LOCAL with SentenceBERT embeddings, and the sample-conditional version of SMOOTHIE-LOCAL estimates $\theta_i(x)$ using a neighborhood size $n_0 = 1$.

Results for the 3B and 7B ensembles over DISTR-ACC and DISTR-ROUGE2 are provided in Table 2. We find that **SMOOTHIE-LOCAL** outperforms all baselines across both data distributions, for both ensembles. Though SMOOTHIE-LOCAL requires no labels, it still outperforms labeled data baselines like LABELED-KNN and PAIRRM. We observe a substantial gap between SMOOTHIE-LOCAL and SMOOTHIE-GLOBAL, which indicates that SMOOTHIE-LOCAL's sample-specific scoring mechanism provides performance improvements.

Method	3B		7B	
	DISTR-ACC	DISTR-ROUGE2	DISTR-ACC	DISTR-ROUGE2
RANDOM	48.7	17.0	65.4	25.0
PAIRRM	53.9	19.0	71.8	25.5
LABELED-KNN	51.0	16.8	71.7	26.2
BEST-MODEL	52.3	18.1	73.2	26.4
SMOOTHIE-GLOBAL	51.3	18.1	66.5	26.1
SMOOTHIE-LOCAL	**58.7**	**20.2**	**75.0**	**26.9**

Table 2. Comparing SMOOTHIE-LOCAL to baseline methods on the 3B and 7B ensembles for multi-task distributions. DISTR-ACC and DISTR-ROUGE2 are measured with accuracy and rouge2 respectively. Bold values indicate the best performing method for each dataset and model size. Metrics are scaled to 0-100.

Notably, we see that SMOOTHIE-LOCAL substantially betters BEST-MODEL, indicating that SMOOTHIE-LOCAL's

routing mechanism is offering a performance improvement over a strategy which merely selects the best LLM on average. We study this in greater detail by examining the relative rank of the LLM selected by SMOOTHIE-LOCAL for each sample. For each sample in DISTR-ACC and DISTR-ROUGE2, we rank the quality of each LLM's generation according to standard-competition ranking (i.e., "1-2-2-4" ranking). We then count how frequently SMOOTHIE-LOCAL selects the rank-i generation across each distribution for each ensemble. We visualize results in Figure 3. As the visualizations demonstrate, SMOOTHIE-LOCAL consistently selects the best or second-best generation from within the ensemble.

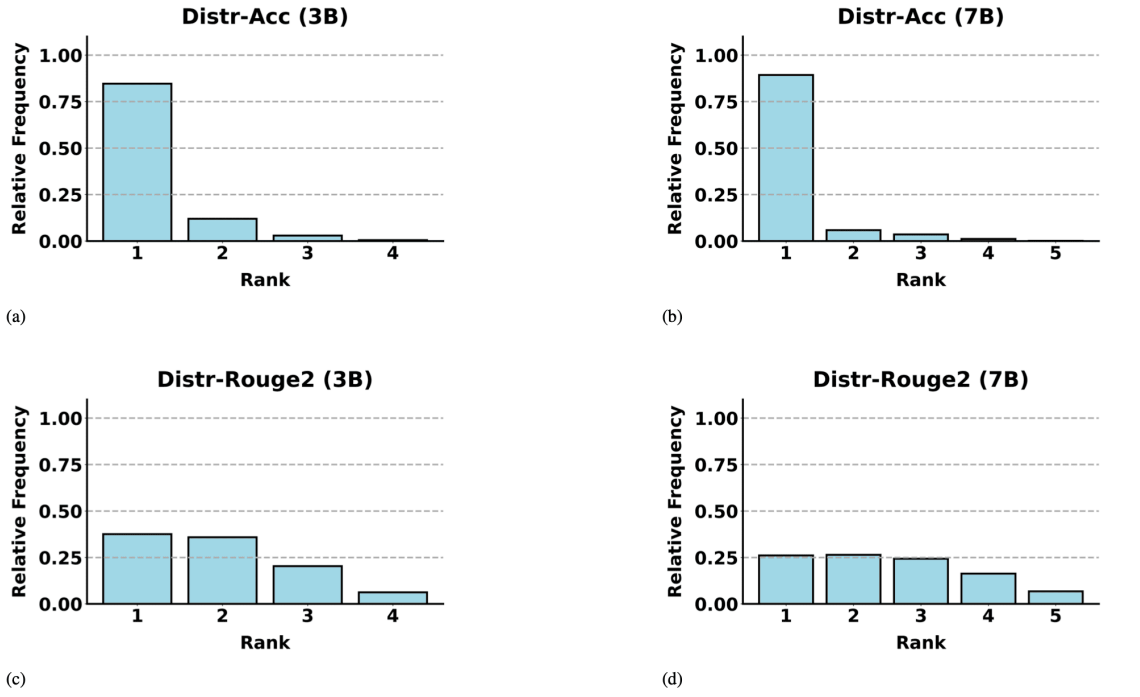

Figure 3. On DISTR-ACC and DISTR-ROUGE2, we measure how frequently SMOOTHIE-LOCAL selects the i-th best generation across the ensemble, for both the 3B and 7B ensembles.

5.3. Prompt Selection

Third, we study whether SMOOTHIE-LOCAL and SMOOTHIE-GLOBAL can be generalized to other settings where engineers have a candidate pool of text generators of unknown quality, and must select one of them to use for some application. In particular, we focus on the setting where an engineer has access to multiple prompt templates for a given generation task, and must select which prompt-templates' generation to use as the final output[59]. Unlike above, we assume the engineer only has access to one LLM. We study SMOOTHIE-LOCAL and SMOOTHIE-GLOBAL in this regime using the NLG tasks from Section 5.1. For each task, we manually write

between 3 and 5 prompt templates, varying the wording of instructions and the choice of in-context samples. We analyze SMOOTHIE applied to two models at different size points: Falcon (1B)[60] and Llama-2 (7B)[48].

Table 3 provides the results. Overall, we find that SMOOTHIE-GLOBAL selects the optimal prompt 2/7 times for Falcon-1B, and 3/7 times for Llama-2. SMOOTHIE-LOCAL and SMOOTHIE-GLOBAL consistently outperform Random—on 6/7 tasks for Falcon-1b and 6/7 tasks for Llama-2. On 7 task/model combinations, one of either SMOOTHIE-GLOBAL or SMOOTHIE-LOCAL matches or outperforms a labeled baseline. To better contextualize performance improvements from SMOOTHIE-GLOBAL, we also compare to the improvement that accompanies increasing model size. Following a common practice in recent work, we can quantify the extent to which SMOOTHIE-GLOBAL allows smaller models to match or exceed the performance of larger models[59][19]. In Figure 4 (Appendix E), we compare Random and SMOOTHIE-GLOBAL on models from the Pythia suite at four sizes: 410M, 1B, 2.8B, and 6.9B parameters[44]. We observe that SMOOTHIE-GLOBAL substantially improves performance—on E2E, SMOOTHIE-GLOBAL enables a 410M parameter model to outperform a 6.9B parameter model.

		CNN	Def. Ext.	E2E	SQuAD	TriviaQA	WebNLG	XSum
Falcon	RANDOM	7.1	60.3	27.8	47.3	22.0	29.2	4.7
	SMOOTHIE-GLOBAL	**7.9**	62.2	**31.6**	**53.3**	**31.4**	28.3	**6.4**
	SMOOTHIE-LOCAL	**8.0**	**69.2**	31.5	**53.3**	27.4	**30.8**	6.0
	BEST-ON-VAL	**8.4**	64.2	31.0	52.7	31.4	**32.5**	**6.7**
Llama-2	RANDOM	7.3	47.8	31.6	54.0	45.9	45.5	11.2
	SMOOTHIE-GLOBAL	6.9	**64.6**	**37.6**	61.4	**68.7**	**48.5**	**12.8**
	SMOOTHIE-LOCAL	**9.5**	59.3	33.6	**63.1**	61.3	48.0	12.7
	BEST-ON-VAL	**11.8**	**64.6**	35.0	**66.1**	**68.7**	**48.7**	**13.0**

Table 3. Comparing SMOOTHIE-GLOBAL and SMOOTHIE-LOCAL to baseline methods in the prompt-selection setting. Underlined values are the best performing *unsupervised* methods. Bold values are the best performing overall methods. We report rouge2 scores for CNN, XSum, WebNLG, and E2E, and accuracy for the rest. All metrics are scaled to 0-100.

5.4. Ablations

Finally, we conduct ablations to examine different aspects of SMOOTHIE-GLOBAL and SMOOTHIE-LOCAL: improving its efficiency, adjusting the neighborhood size, varying the choice of embedding model, and using different LLM ensembles.

Improving efficiency

First, we explain SMOOTHIE's current efficiency properties. To estimate the SMOOTHIE weights for routing, we use a simple closed-form procedure that does not require any SGD or training, as described in Algorithm 1. As a result, SMOOTHIE weights on the entire dataset can be computed in seconds—for the 7B ensemble, SMOOTHIE-LOCAL on the multi-task datasets takes 2.14 seconds per 1000 samples, and SMOOTHIE-GLOBAL on the single-task datasets takes under 0.03 seconds per 1000 samples. Moreover, SMOOTHIE does not require any ground-truth annotations; however, all m model generations per test sample are needed as input to the algorithm. That is, we need $n \times m$ generations for a $\mathcal{D}_{\text{train}}$ of size n samples.

Fortunately, the need for computing all model generations per test sample can be removed with a small algorithm tweak, making SMOOTHIE even more efficient and its runtime independent of n. Suppose we have a held-out set of n_{train} train samples with precomputed generations from the models in the ensemble. For each test sample, we retrieve the most similar train samples, learn the SMOOTHIE weights for the sample using the corresponding train sample generations, and return the model with the highest SMOOTHIE weight (i.e., in line 5 in Algorithm 1, KNN is now over a held-out training dataset). This approach, which we call SMOOTHIE-Train, selects the model for a test sample without needing model generations for that sample. Only $n_{\text{train}} \times m$ generations are needed, regardless of how large the test dataset n is.

We study the NLG tasks, using $n_{\text{train}} = 250$ samples. In Table 7 (Appendix E), we evaluate a version of SMOOTHIE-GLOBAL-Train) and observe that it matches SMOOTHIE-GLOBAL on 12/14 model-dataset pairs, and performs worse on the remaining 2/14 pairs. We also evaluate SMOOTHIE-LOCAL-Train, on DISTR-ACC and DISTR-ROUGE2 (Table 8) using a neighborhood of size $n_0 = 20$. We find here that while SMOOTHIE-LOCAL-Train underperforms SMOOTHIE-LOCAL on both the 3B and 7B ensemble for both DISTR-ACC and DISTR-ROUGE2, it still outperforms Random and remains competitive with supervised baselines.

Neighborhood size

We study the impact of n_0, and consider SMOOTHIE-LOCAL's performance for $n_0 \in [1, 5, 10, 20, 50, 100]$. Figure 5 provides performance over DISTR-ACC and Figure 6 provides performance over DISTR-ROUGE2. Overall, we find that SMOOTHIE-LOCAL's performance steadily degrades as n_0 increases, and is highest when $n_0 = 1$.

Choice of embeddings

We study how the choice of embeddings affects SMOOTHIE-GLOBAL performance (Table 9). Specifically, we compare the performance of SMOOTHIE-LOCAL using Sentence-Bert embeddings (`all-mpnet-base-v2`) [33] to BGE embeddings (`bge-small-en-v1.5`) [61]. We observe that SMOOTHIE-LOCAL appears robust to different embeddings—SMOOTHIE-LOCAL with BGE embeddings still outperforms other labeled and unlabeled baselines. Interestingly, we observe that certain embedding models appear to yield better performance over certain

distribution/ensemble combinations. For instance, SMOOTHIE-LOCAL with SentenceBERT embeddings outperforms SMOOTHIE-LOCAL with BGE embeddings on DISTR-ACC for the 3B ensemble and DISTR-ROUGE2 for the 7B ensemble, while performing worse on DISTR-ROUGE2 for the 3B ensemble and DISTR-ACC for the 7B ensemble.

Different ensembles

Finally, we consider whether SMOOTHIE-GLOBAL can generalize to a wider array of ensembles (Figure 7). We combine the LLMs contained in the 3B and 7B ensembles into a single pool, and sample 50 distinct ensembles ranging in size from 4-7 LLMs. For each of the 7 NLG tasks, we evaluate SMOOTHIE-GLOBAL's ability to identify the best model from within each ensemble. Across these 350 settings, we find that SMOOTHIE-GLOBAL identifies the best model in 211 of them (60.2 of the time), and one of the two best models in 292 of them (83 of the time).

6. Conclusion

In this paper we study and propose an algorithm for learning label-free routers for generative tasks. We validate our approach across a variety of evaluation regimes, finding it consistently beats other unsupervised approaches and often matches/exceeds supervised approaches.

Limitations

We discuss several of SMOOTHIE's limitations. First, its multivariate Gaussian graphical model currently uses a diagonal covariance matrix. This assumes independent error vectors for each generation, though SMOOTHIE could be extended to account for dependencies[31][62]. Additionally, SMOOTHIE optimizes only for performance without considering cost tradeoffs between large and small models. Finally, its reliance on embeddings may capture only certain aspects of semantic similarity. Other embedding models and additional heuristics could be used to create richer input features for SMOOTHIE.

Appendix A.

In Appendix B, we provide a glossary of notation used in the paper. In Appendix C, we provide an extended related work, and in Appendix D we provide a proof of Proposition 1, which is used in deriving the SMOOTHIE algorithm. Finally, in Appendix E we provide additional experimental results and details.

Code for reproducing our results and using SMOOTHIE is available at https://github.com/HazyResearch/SMOOTHIE.

Appendix B. Notation

The glossary is given in Table 4 below.

Symbol	Used for
$\overline{\mathcal{V}}$	The space of all vocabulary sequences.
x	Input text $x \in \mathcal{X} \subset \overline{\mathcal{V}}$.
y	Reference output text $y \in \mathcal{Y} \subset \overline{\mathcal{V}}$.
G	Candidate pool of m LLMs, $G = \{g_1, \ldots, g_m\}$, where each $g_i \in \mathcal{G} : \mathcal{X} \to \mathcal{Y}$
	produces a generation $g_i(x)$ on input x.
$\mathcal{D}_{\text{test}}$	Unlabeled test dataset $\mathcal{D}_{\text{test}} = \{x_i\}_{i=1}^{n}$.
$route$	Routing function $route : \mathcal{G}^m \times \mathcal{X} \to \mathcal{G}$ that selects the best LLM from G for each sample.
$\theta_i(x)$	Quality score of the ith LLM on test sample x, also used in the graphical model in (1).
z_{g_0}	Embedding mapping $z_{g_0} : \overline{\mathcal{V}} \to \mathbb{R}^d$ for any text sequence, where g_0 is an embedding model
	such as SentenceBERT [33].
$\lambda_i(x)$	The observable embedding of x concatenated with the ith LLM's generated output,
	$\lambda_i(x) := z_{g_0}([x, g_i(x)])$.
$z^\star(x)$	The latent embedding of x concatenated with unknown reference output, $z^\star(x) := z_{g_0}([x, y])$.
Z	Partition function for normalization of (1).
n_0	Number of nearest neighbors used to learn $\theta_i(x)$ for x. $n_0 = n$ (i.e., the entire test dataset)
	corresponds to SMOOTHIE-GLOBAL and $n_0 < n$ corresponds to SMOOTHIE-LOCAL.
$\hat{\delta}_{ij}(x)$	The average squared Euclidean distance between the ith and jth LLM embeddings over
	a neighborhood around x, $\hat{\delta}_{ij}(x) = \frac{1}{n_0} \sum_{x' \in \text{NN}_{n_0}(x)} \|\lambda_i(x') - \lambda_j(x')\|^2$.
	This is the primary expression used in computing $\theta_i(x)$.

Table 4. Glossary of variables and symbols used in this paper.

Appendix C. Extended Related Work

LLM Routing

The problem of determining how to route samples to various models has been long studied in statistics[11][12] as well as Mixture of Experts deep neural networks[14][13]. These works focus on how to jointly train the models and router in a stable and efficient manner.

Since many LLMs are now available off-the-shelf, recent works study how routing mechanisms can be applied at inference time to trained models. Some works involve training task or domain-specific expert models and then learning a router. The router can be a nearest neighbors algorithm[15], a neural network[63] that classifies among the different domains corresponding to the experts, or an extra gate learned when training the expert models[64]. These approaches do not explicitly require labels, but they require knowledge of what domain is used to train each expert and assume that each expert is the best model for its corresponding domain, therefore effectively using this mapping as a form of labels. In contrast, our setting focuses on routing among pre-trained LLMs where we do not know what models are optimal on what tasks and their samples.

A second category of inference-time routing works studies how to choose among a collection of pre-trained LLMs, which is the setting that SMOOTHIE focuses on. Several approaches involve training a meta-model that either scores or ranks how a LLM will perform on a sample[8][6][16], all of which required labeled data to train. MoRE[65] involves training a simpler random forest classifier, using the rate of agreement among LLMs as one of the features, which is similar to how SMOOTHIE estimates scores; however, it also requires labeled data to train the classifier. Some approaches[17][5] do not require training routers and simply use nearest neighbor methods. However, these nearest neighbor methods still use labeled data to determine what training samples each LLM performs the best on.[7] invokes a trained reward model for the routing mechanism.[66] trains a classification-based router using the BARTScore metric on LLM generations as pseudolabels; this avoids using manually labeled data, demonstrating that while a majority of routing methods require labeled data, there exist some alternatives that do not. We leave it to future work to compare and integrate SMOOTHIE with other unsupervised approaches.

Finally, complementary to our setting are works that jointly focus on cost minimization as well as quality of generations. RouterBench[67] creates a benchmark for studying the cost-quality tradeoffs in routing systems. Optimizing for cost can be done algorithmically, such as in FrugalGPT[68], AutoMix[69], RouteLLM[70], and[71], as well as via hardware enhancements such as SambaNova Systems' Composition of Experts[72].

LLM Ensembling

A rich literature has observed that ensembling LLM outputs—across different prompts or base models—can improve the accuracy of generated predictions. Prior work has proposed and studied a number of different ensembling algorithms for classification tasks, including majority-voting[73][18], weak-supervision[59][19], boosting[20][74][75], and others[76][77][78].

More relevant to our work here is a literature on ensembling for generative tasks. One category of methods rely on an auxilliary sequence-to-sequence models to "fuse" generations from different prompts or base LLMs[6]. Though recently applied in the context of modern LLMs, the concept of fusion traces back to older work on summarization[79][80][81][82]. Some techniques combine or switch among multiple outputs at inference time[21]

[83][22][84][85], while others involve averaging in weight space[15][86][23]. Lastly, ensembling can also be approximated by randomly selecting a model to be used in multi-turn settings[58].

Other LLM Selection Algorithms

Beyond the setting of selecting among multiple LLMs, other works have explored how to select the optimal prompt template from a collection of candidate prompts. These works can be grouped into two categories. The first category assumes that engineers have access to labeled data. In the naive case, this labeled data can simply be used to select the best performing prompt[24][87][88]. Another subset of this category focuses on the setting where new prompts can be generated by selecting in-context demonstrations from a set of labeled samples (typically a small training set)[89][90]. Prior work has proposed different methods for identifying the optimal in-context demonstrations to use, depending on the sample for which the LLM is being used to produce a prediction for[91][88][25][92][93][94]. The second category focuses on zero-label prompt selection methods, but solely for classification tasks[95][26][27]. Prior work here selects prompts on the basis of mutual information[26], agreement rates between predictions produced by different prompts[95], and various probability based measures[27][96][97].

Weak supervision

SMOOTHIE utilizes techniques inspired by weak supervision literature. Weak supervision aims to programmatically generate labels on an unlabeled dataset by aggregating the predictions of several weak "voters", such as heuristics, pre-trained models, and knowledge graphs[28][29]. It assumes a particular latent variable graphical model and uses its structure to estimate latent quantities, such as the accuracy of each voter (in our setting, the quality score of each LLM). Typically, this graphical model is a binary Ising model, as weak supervision has generally been studied in classification settings[31][30], where embeddings have been utilized as auxiliary signal but not modeled explicitly[59][98]. Weak supervision has been applied to broader settings, such as for learning rankings, graphs, and manifolds[9][32]. We derive our estimation procedure from the Gaussian model in[9], applying it to LLM embeddings. While both SMOOTHIE and[9] use a multivariate Gaussian model, in SMOOTHIE we apply it to model routing with SBERT embeddings on natural language datasets, whereas[9] conducts synthetic experiments in hyperbolic spaces and metric spaces induced by synthetic graphs. Moreover, SMOOTHIE uses nearest neighbor kernel smoothing to allow for sample-dependent weights—critical for routing—while[9] calculates one global set of weights over the dataset.

Consistency-based selection

Consistency is central to unsupervised selection and aggregation methods, the simplest being majority vote. While weak supervision methods[30] and SMOOTHIE heavily rely on notions of voter agreement as depicted in a graphical model, there are several other consistency-based methods. Minimum Bayes Risk methods[99]

[100] selects the generation that has the highest average similarity (i.e., cosine) with other generations. This is similar to SMOOTHIE, which routes to the lowest value of (3). If we ignore the subtraction of $\delta_{jk}(x)$ in (3) and average over more than just $\delta_{ij}(x)$ and $\delta_{ik}(x)$, then SMOOTHIE with $n_0 = 1$ is equivalent to[99] Therefore, SMOOTHIE can be considered as a slightly modified and more general version of this approach. Another approach[101] relies on consistency between a "global" and "local" embedding for each generation. They solve an optimization problem that estimates each generation's quality score by constructing a loss that enforces that the similarity between the estimated true generation (produced by a weighted average of candidates) and the candidate generation should be the same according to both global and local embeddings. In contrast, SMOOTHIE uses one embedding space, relies on a multivariate Gaussian structure among embeddings, and does not require gradient descent to learn the quality of each generation.

Test-Time Compute

Approaches like model routing, ensembling, and selection can all be seen as ways of utilizing *test-time compute* to produce higher-quality generations from a system of LLMs. Test-time compute can also be utilized over a single LLM via techniques such as those used in OpenAI's o1, Chain of Thought, and Rephrase and Respond[102][103][104]. Other works have recently studied how test-time compute scales[105][106]—finding that producing more generations can often yield the correct response—and how to combine multiple test-time methods, such as Archon[107]. It is interesting future work to consider how SMOOTHIE can be integrated with other test-time compute techniques.

Appendix D. Proof of Proposition 1

We provide a proof of proposition 1, which is a direct property of multivariate Gaussians that is also presented in[9]. We first expand $\mathbb{E}[\|\lambda_i(x) - \lambda_j(x)\|^2]$:

$$\begin{aligned}\mathbb{E}[\|\lambda_i(x) - \lambda_j(x)\|^2] &= \mathbb{E}[\|(\lambda_i(x) - z^*(x)) - (\lambda_j(x) - z^*(x))\|^2] \\ &= \mathbb{E}[\|\lambda_i(x) - z^*(x)\|^2] + \mathbb{E}[\|\lambda_j(x) - z^*(x)\|^2] - 2\mathbb{E}[(\lambda_i(x) - z^*(x))^\top(\lambda_j(x) - z^*(x))]\end{aligned} \tag{5}$$

Let $\lambda_{i,k}(x)$ be the kth element of the $\lambda_i(x)$ embedding, and similarly define $z^*_k(x)$. Note that since Σ is diagonal, we can write

$$\begin{aligned}&\mathbf{Cov}[\lambda_{i,k}(x) - z^*_k(x), \lambda_{j,k}(x) - z^*_k(x)] \\ &= \mathbb{E}[(\lambda_{i,k}(x) - z^*_k(x)) \cdot (\lambda_{j,k}(x) - z^*_k(x))] - \mathbb{E}[\lambda_{i,k}(x) - z^*_k(x)]\mathbb{E}[\lambda_{j,k}(x) - z^*_k(x)] \\ &= 0\end{aligned}$$

for all $k \in [d]$. Since $\mu = \vec{0}$, we thus have that $\mathbb{E}[(\lambda_{i,k}(x) - z^*_k(x)) \cdot (\lambda_{j,k}(x) - z^*_k(x))] = 0$ for all $k \in [d]$, which implies that $\mathbb{E}[(\lambda_i(x) - z^*(x))^\top(\lambda_j(x) - z^*(x))] = 0$. Plugging this into (5), we have

$$\mathbb{E}[\|\lambda_i(x) - \lambda_j(x)\|^2] = \mathbb{E}[\|\lambda_i(x) - z^*(x)\|^2] + \mathbb{E}[\|\lambda_j(x) - z^*(x)\|^2]. \tag{6}$$

Appendix E. Additional Experiments and Details

This section contains additional details on experiments discussed in Section 5.

E.1. Datasets and Models

Table 5 provides links to the Huggingface datasets used for each task. For E2E, CNN/DailyMail, XSum, and Web NLG we measure performance using rouge2. For SQuAD, TriviaQA, and Definition Extraction we measure using "accuracy." A model generation is treated as "correct" if if contains the answer, and incorrect otherwise[39].

Dataset name	Huggingface URL
E2E	https://huggingface.co/datasets/e2e_nlg
CNN/DailyMail	https://huggingface.co/datasets/cnn_dailymail
SQuAD	https://huggingface.co/datasets/hazyresearch/based-squad
XSum	https://huggingface.co/datasets/EdinburghNLP/xsum
TriviaQA	https://huggingface.co/datasets/mandarjoshi/trivia_qa
Web NLG	https://huggingface.co/datasets/web_nlg
Definition Extraction	https://huggingface.co/datasets/nguha/legalbench

Table 5. Datasets used.

Table 6 contains links for all models used.

Model name	Huggingface URL
Pythia-410M	https://huggingface.co/EleutherAI/pythia-410m
Pythia-1B	https://huggingface.co/EleutherAI/pythia-1b
Pythia-2.8B	https://huggingface.co/EleutherAI/pythia-2.8b
Pythia-6.9B	https://huggingface.co/EleutherAI/pythia-6.9b
Gemma-2B	https://huggingface.co/google/gemma-2b-it
Incite-3B	https://huggingface.co/togethercomputer/RedPajama-INCITE-Instruct-3B-v1
Dolly-3B	https://huggingface.co/databricks/dolly-v2-3b
Llama-2-7B	https://huggingface.co/meta-llama/Llama-2-7b-hf
Mistral-7B	https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
Vicuna-7B	https://huggingface.co/lmsys/vicuna-7b-v1.5
Gemma-7B	https://huggingface.co/google/gemma-7b
Nous Capybara	https://huggingface.co/NousResearch/Nous-Capybara-7B-V1.9
Phi-2	https://huggingface.co/microsoft/phi-2
Llema-7B	https://huggingface.co/EleutherAI/llemma_7b

Table 6. Huggingface model URLs.

For the Alpaca leaderboard experiments, we run each trial by sampling 5 models from the following set of 10: `Nanbeige-Plus-Chat-v0.1`, `claude-2`, `Qwen1.5-110B-Chat`, `yi-large-preview`, `gemini-pro`, `Meta-Llama-3-70B-Instruct`, `Ein-70B-v0.1`, `mistral-large-2402`, `Storm-7B`, `FsfairX-Zephyr-Chat-v0.1`.

E.2. Additional Results

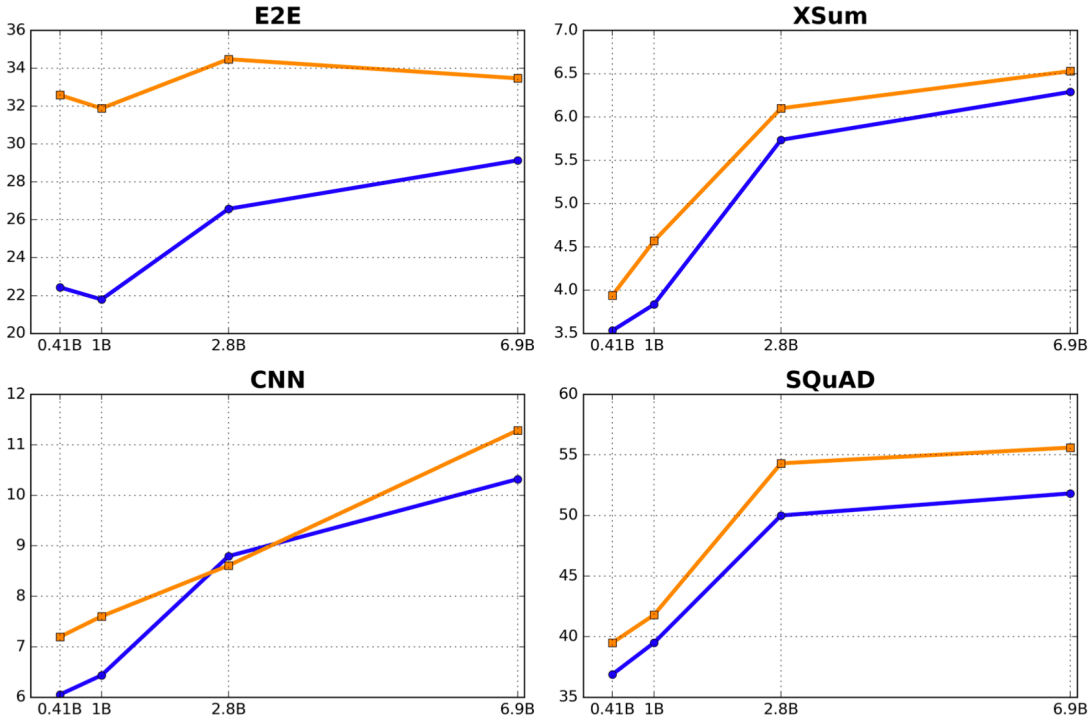

Figure 4. We compare RANDOM (blue) and SMOOTHIE-GLOBAL (orange) for prompt-selection on different sized models in the Pythia suite. The x-axis denotes model size, and the y-axis denotes performance (either rouge2 or accuracy).

		CNN	Def. Ext.	E2E	SQuAD	TriviaQA	WebNLG	XSum
3B	RANDOM	12.9	52.4	27.3	59.6	32.7	23.4	4.5
	SMOOTHIE-GLOBAL	**14.3**	**61.5**	**31.8**	60.7	32.1	**30.7**	4.5
	SMOOTHIE-GLOBAL-Train	**14.3**	**61.5**	24.7	**60.7**	32.1	**30.7**	4.5
	BEST-ON-VAL	13.0	60.5	31.1	**66.4**	**38.7**	30.3	**5.3**
7B	RANDOM	13.7	58.5	35.3	67.9	59.3	44.1	6.9
	SMOOTHIE-GLOBAL	**14.5**	**70.9**	**36.9**	**76.2**	**68.3**	45.9	**8.4**
	SMOOTHIE-GLOBAL-Train	**14.5**	**70.9**	36.5	**76.2**	**68.3**	45.9	**8.4**
	BEST-ON-VAL	**14.5**	69.4	36.7	74.0	65.8	**48.3**	8.3

Table 7. We compare SMOOTHIE-GLOBAL to SMOOTHIE-GLOBAL-Train, for which weights are learned on a hold-out set. We provide results from baseline methods for reference. Underlined values are the best performing *unsupervised* methods. Bold values are the best performing *overall* methods. We report rouge2 scores for CNN, XSum, WebNLG, and E2E, and accuracy for the rest. All metrics are scaled to 0-100.

doi.org/10.32388/F319W4

Method	3B		7B	
	DISTR-ACC	DISTR-ROUGE2	DISTR-ACC	DISTR-ROUGE2
RANDOM	48.7	17.0	65.4	25.0
PAIRRM	53.9	19.0	71.8	25.5
LABELED-KNN	51.0	16.8	71.7	26.2
BEST-MODEL	52.3	18.1	73.2	26.4
SMOOTHIE-GLOBAL	51.3	18.1	66.5	26.1
SMOOTHIE-LOCAL	**58.7**	**20.2**	**75.0**	**26.9**
SMOOTHIE-GLOBAL-train	51.3	18.1	66.5	26.1
SMOOTHIE-LOCAL-train	50.7	18.8	70.9	26.0

Table 8. We compare SMOOTHIE-LOCAL to SMOOTHIE-LOCAL-train, for which weights are learned on a hold-out set, on the 3B and 7B ensembles for multi-task distributions. DISTR-ACC and DISTR-ROUGE2 are measured with accuracy and rouge2 respectively. Bold values indicate the best performing method for each dataset and model size. Metrics are scaled to 0-100. Other baseline methods are provided for comparison.

Method	3B		7B	
	DISTR-ACC	DISTR-ROUGE2	DISTR-ACC	DISTR-ROUGE2
RANDOM	48.7	17.0	65.4	25.0
PAIRRM	53.9	19.0	71.8	25.5
LABELED-KNN	51.0	16.8	71.7	26.2
BEST-MODEL	52.3	18.1	73.2	26.4
SMOOTHIE-LOCAL (BGE-small[61])	**59.3**	19.7	74.6	**27.1**
SMOOTHIE-LOCAL (SBERT[33])	58.7	**20.2**	**75.0**	26.9

Table 9. Comparing SMOOTHIE-LOCAL with different embeddings on the 3B and 7B ensembles for multi-task distributions. DISTR-ACC and DISTR-ROUGE2 are measured with accuracy and rouge2 respectively. Bold values indicate the best performing method for each dataset and model size. Metrics are scaled to 0-100.

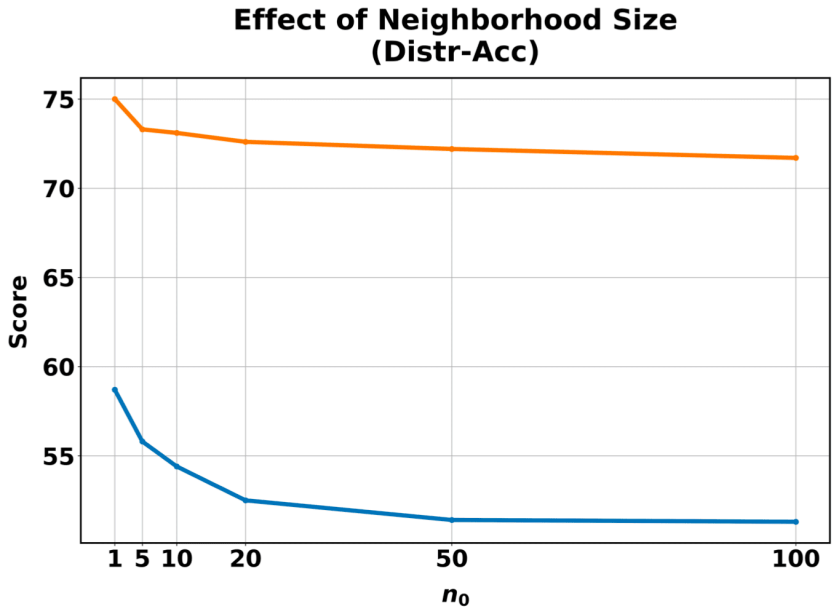

Figure 5. We measure how SMOOTHIE-LOCAL's performance on DISTR-ACC changes as n_0 changes.

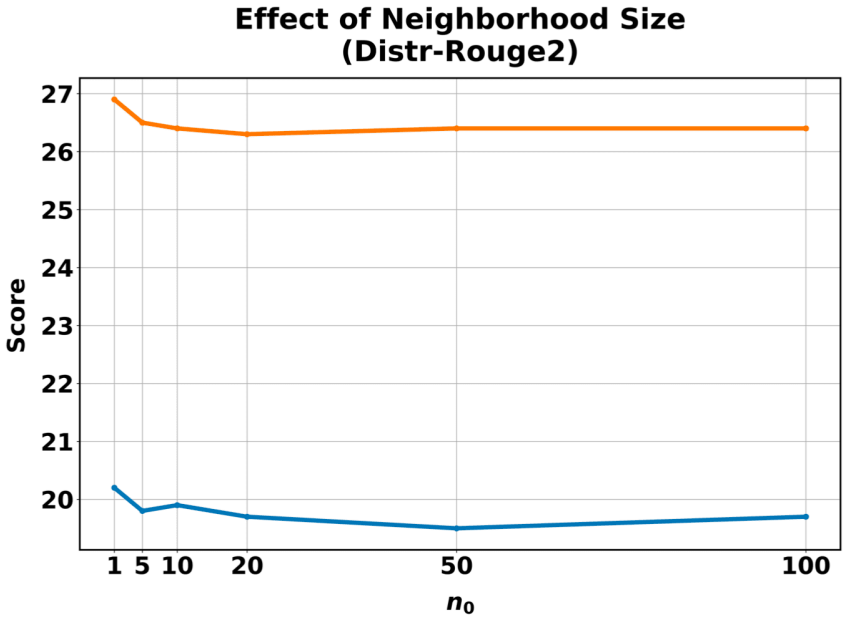

Figure 6. We measure how SMOOTHIE-LOCAL's performance on DISTR-ROUGE2 changes as n_0 changes.

doi.org/10.32388/F319W4

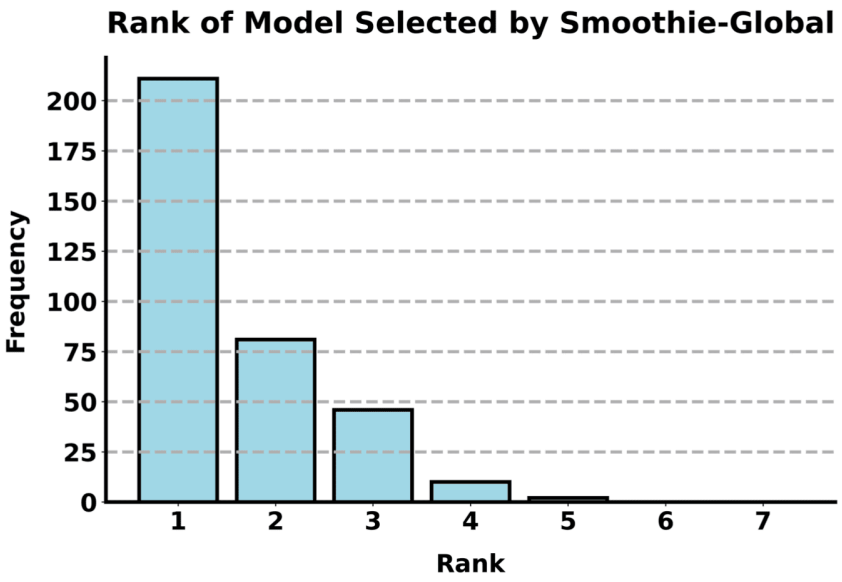

Figure 7. We evaluate SMOOTHIE-GLOBAL's ability to identify the best model by randomly sampling 50 ensembles of size 4-7 LLMs from a pool of the LLMs contained in the 3B and 7B ensembles. We apply SMOOTHIE-GLOBAL to select the best LLM from within each of these ensembles across the 7 NLG tasks, and measure the rank (relative to the ensemble) of the LLM selected by SMOOTHIE-GLOBAL.

Method	ChatGPT-Rank (↓)
RANDOM	5.96
SMOOTHIE-GLOBAL	3.91

Table 10. Results for SMOOTHIE-GLOBAL and baselines on MixInstruct.

Method	Accuracy
RANDOM	28.4
BEST-ON-VAL	37.5
SMOOTHIE-GLOBAL	37.5

Table 11. Results for SMOOTHIE-GLOBAL and baselines on GSM8K. We report accuracy, with scores scaled to 0-100.

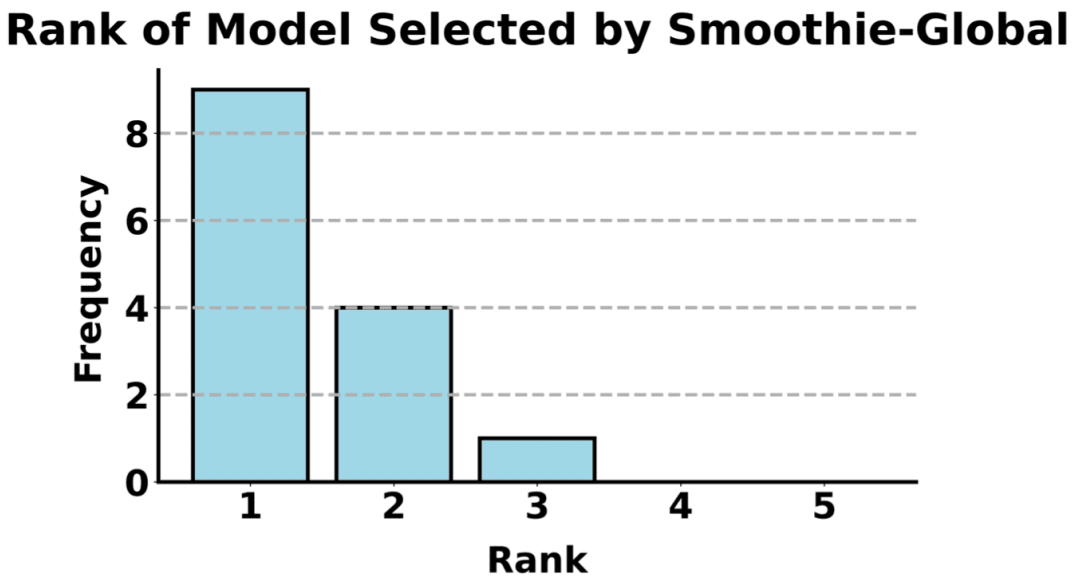

Figure 8. We construct a histogram over the rank of the LLM selected by SMOOTHIE-GLOBAL across both the 3B and 7B ensembles, for 7 NLG tasks.

Acknowledgements

We thank Gautam Machiraju, Jon Saad-Falcon, Krista Opsahl-Ong, Sabri Eyuboglu, Jordan Juravsky, Vishnu Sarukkai, Ben Spector, Eric Nguyen, Jerry Liu, Chris Fifty, Avanika Narayan, Michael Zhang, Vincent Chen, and Fred Sala for their helpful feedback and discussion. This research project has benefitted from the Microsoft Accelerate Foundation Models Research (AFMR) grant program.

We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF2247015 (Hardware-Aware), CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); US DEVCOM ARL under Nos. W911NF-23-2-0184 (Long-context) and W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under Nos. N000142312633 (Deep Signal Processing); Stanford HAI under No. 247183; NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Meta, Google, and VMWare. NG is suported by the Stanford Interdisciplinary Graduate Fellowship (SIGF). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views, policies, or endorsements, either expressed or implied, of NIH, ONR, or the U.S. Government.

Footnotes

[1] Definition Extraction has fewer than 1000 samples.

[2] Responses are available on the AlpacaEval website: https://tatsu-lab.github.io/alpaca_eval/.

References

1. ^a ^b *Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, Bernstein MS, Bohg J, Bosselut A, Brunskill E, et al. (2021). "On the opportunities and risks of foundation models". arXiv preprint arXiv:2108.07258.* ***2021****. Available f rom: https://arxiv.org/abs/2108.07258.*
2. ^ *Arora S, Yang B, Eyuboglu S, Narayan A, Hojel A, Trummer I, Ré C (2023). "Language Models Enable Simple Syste ms for Generating Structured Views of Heterogeneous Data Lakes". arXiv.* ***cs.CL****. Available from: https://arxiv.org/ab s/2304.09433.*
3. ^ *Chen M, Tworek J, Jun H, Yuan Q, Pinto HP de Oliveira, Kaplan J, Edwards H, Burda Y, Joseph N, Brockman G, et al. (2021). "Evaluating large language models trained on code". arXiv preprint arXiv:2107.03374.*
4. ^a ^b *Guha N, Nyarko J, Ho D, R{\'e} C, Chilton A, Chohlas-Wood A, Peters A, Waldon B, Rockmore D, Zambrano D, et a l. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models. Advances in Neural Information Processing Systems.* ***36****, 2024.*
5. ^a ^b ^c ^d ^e ^f *Shnitzer T, Ou A, Silva M, Soule K, Sun Y, Solomon J, Thompson N, Yurochkin M (2023). "Large Language Model Routing with Benchmark Datasets". arXiv. arXiv:2309.15789 [cs.CL].*
6. ^a ^b ^c ^d ^e ^f ^g ^h ^i ^j *Jiang D, Ren X, Lin BY. "LLM-Blender: Ensembling Large Language Models with Pairwise Compa rison and Generative Fusion." In: Proceedings of the 61th Annual Meeting of the Association for Computational Ling uistics (ACL 2023); 2023.*
7. ^a ^b *Lu K, Yuan H, Lin R, Lin J, Yuan Z, Zhou C, Zhou J (2023). "Routing to the Expert: Efficient Reward-guided Ensem ble of Large Language Models". arXiv. arXiv:2311.08692 [cs.CL].*
8. ^a ^b ^c *Sakota M, Peyrard M, West R. Fly-Swat or Cannon? Cost-Effective Language Model Choice via Meta-Modelin g. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining, WSDM '24. ACM; 2024 Mar. doi:10.1145/3616855.3635825.*
9. ^a ^b ^c ^d ^e ^f ^g ^h ^i ^j ^k *Shin C, Li W, Vishwakarma H, Roberts NC, Sala F (2022). "Universalizing Weak Supervision". I n: International Conference on Learning Representations. Available from: https://openreview.net/forum?id=YpPiNig TzMT.*
10. ^a ^b *Li X, Zhang T, Dubois Y, Taori R, Gulrajani I, Guestrin C, Liang P, Hashimoto TB. AlpacaEval: An Automatic Evalu ator of Instruction-following Models.* ***GitHub repository****. 2023. Available from: https://github.com/tatsu-lab/alpaca _eval.*

11. a, b *Jordan MI, Jacobs RA (1993). "Hierarchical mixtures of experts and the EM algorithm". In: Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan). 2: 1339-1344 vol.2. doi:10.1109/IJCNN.1993.716791.*
12. a, b *Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991). "Adaptive Mixtures of Local Experts". Neural Computation. 3 (1): 79–87. doi:10.1162/neco.1991.3.1.79.*
13. a, b *Shazeer N, Mirhoseini A, Maziarz K, Davis A, Le Q, Hinton G, Dean J (2017). "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer". arXiv. arXiv:1701.06538 [cs.LG].*
14. a, b *Fedus W, Zoph B, Shazeer N (2022). "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity". arXiv. arXiv:2101.03961 [cs.LG].*
15. a, b, c *Jang J, Kim S, Ye S, Kim D, Logeswaran L, Lee M, Lee K, Seo M (2023). "Exploring the Benefits of Training Expert Language Models over Instruction Tuning". arXiv. arXiv:2302.03202.*
16. a, b *Ravaut M, Joty S, Chen NF (2023). "SummaReranker: A Multi-Task Mixture-of-Experts Re-ranking Framework for Abstractive Summarization". arXiv. arXiv:2203.06569 [cs.CL].*
17. a, b, c *Lee CH, Cheng H, Ostendorf M (2023). "OrchestraLLM: Efficient Orchestration of Language Models for Dialogue State Tracking". arXiv. arXiv:2311.09758.*
18. a, b *Wang X, Wei J, Schuurmans D, Le Q, Chi E, Narang S, Chowdhery A, Zhou D (2022). "Self-consistency improves chain of thought reasoning in language models". arXiv preprint arXiv:2203.11171.*
19. a, b, c *Arora S, Narayan A, Chen MF, Orr L, Guha N, Bhatia K, Chami I, Sala F, R\u00e9 C (2022). "Ask Me Anything: A simple strategy for prompting language models". arXiv. arXiv:2210.02441 [cs.CL].*
20. a, b *Pitis S, Zhang MR, Wang A, Ba J (2023). "Boosted prompt ensembles for large language models". arXiv preprint arXiv:2304.05970. **2023**.*
21. a, b *Izacard G, Grave E (2020). "Leveraging passage retrieval with generative models for open domain question answering". arXiv preprint arXiv:2007.01282.*
22. a, b *Shen SZ, Lang H, Wang B, Kim Y, Sontag D (2024). "Learning to Decode Collaboratively with Multiple Language Models". arXiv. arXiv:2403.03870 [cs.CL].*
23. a, b *Wan F, Yang Z, Zhong L, Quan X, Huang X, Bi W (2024). "FuseChat: Knowledge Fusion of Chat Models". arXiv. arXiv:2402.16107.*
24. a, b, c *Perez E, Kiela D, Cho K (2021). "True few-shot learning with language models". Advances in neural information processing systems. 34: 11054–11070.*
25. a, b *Su H, Kasai J, Wu CH, Shi W, Wang T, Xin J, Zhang R, Ostendorf M, Zettlemoyer L, Smith NA, et al. (2022). "Selective annotation makes language models better few-shot learners". arXiv preprint arXiv:2209.01975.*
26. a, b, c *Sorensen T, Robinson J, Rytting CM, Shaw AG, Rogers KJ, Delorey AP, Khalil M, Fulda N, Wingate D (2022). "An information-theoretic approach to prompt engineering without ground truth labels". arXiv preprint arXiv:2203.11364. Available from: https://arxiv.org/abs/2203.11364.*

27. a, b, cYang S, Kim J, Jang J, Ye S, Lee H, Seo M (2023). "Improving probability-based prompt selection through unifie d evaluation and analysis". arXiv preprint arXiv:2305.14877.
28. a, bRatner A, De Sa C, Wu S, Selsam D, Ré C (2017). "Data Programming: Creating Large Training Sets, Quickly". arXi v. **stat.ML**. Available from: arXiv:1605.07723.
29. a, bRatner A, Bach SH, Ehrenberg H, Fries J, Wu S, Ré C. Snorkel: rapid training data creation with weak supervision. Proceedings of the VLDB Endowment. **11**(3):269–282, 2017. doi:10.14778/3157794.3157797.
30. a, b, cFu D, Chen M, Sala F, Hooper S, Fatahalian K, Re C. "Fast and Three-rious: Speeding Up Weak Supervision with Triplet Methods." In: Daumé III H, Singh A, editors. Proceedings of the 37th International Conference on Machine Le arning. PMLR; 2020. p. 3280-3291. Available from: https://proceedings.mlr.press/v119/fu20a.html. PDF: http://proc eedings.mlr.press/v119/fu20a/fu20a.pdf.
31. a, b, cRatner A, Hancock B, Dunnmon J, Sala F, Pandey S, Ré C (2018). "Training Complex Models with Multi-Task W eak Supervision". arXiv. arXiv:1810.02840 [stat.ML].
32. a, bVishwakarma H, Roberts N, Sala F (2022). "Lifting Weak Supervision To Structured Prediction". arXiv. arXiv:2211. 13375 [cs.LG].
33. a, b, c, d, e, fReimers N, Gurevych I (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". I n: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Comp utational Linguistics. Available from: http://arxiv.org/abs/1908.10084.
34. ^Liang P, Bommasani R, Lee T, Tsipras D, Soylu D, Yasunaga M, Zhang Y, Narayanan D, Wu Y, Kumar A, Newman B, Yuan B, Yan B, Zhang C, Cosgrove C, Manning CD, R\u00e9 C, Acosta-Navas D, Hudson DA, Zelikman E, Durmus E, L adhak F, Rong F, Ren H, Yao H, Wang J, Santhanam K, Orr L, Zheng L, Yuksekgonul M, Suzgun M, Kim N, Guha N, Ch atterji N, Khattab O, Henderson P, Huang Q, Chi R, Xie SM, Santurkar S, Ganguli S, Hashimoto T, Icard T, Zhang T, C haudhary V, Wang W, Li X, Mai Y, Zhang Y, Koreeda Y. Holistic evaluation of language models. 2023. arXiv. Availabl e from: https://arxiv.org/abs/2211.09110.
35. ^Hermann KM, Kociský T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P. Teaching machines to read an d comprehend. In: NIPS; 2015. p. 1693-1701. Available from: http://papers.nips.cc/paper/5945-teaching-machines-t o-read-and-comprehend.
36. ^See A, Liu PJ, Manning CD. Get to the point: Summarization with pointer-generator networks. In: Proceedings of th e 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Vancouver, Canad a: Association for Computational Linguistics; 2017. p. 1073-1083. doi:10.18653/v1/P17-1099. https://www.aclweb.or g/anthology/P17-1099.
37. ^Narayan S, Cohen SB, Lapata M (2018). "Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization". arXiv. arXiv:1808.08745.
38. ^Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Tex t". arXiv. arXiv:1606.05250 [cs.CL].

39. ^a, ^b*Arora S, Eyuboglu S, Zhang M, Timalsina A, Alberti S, Zinsley D, Zou J, Rudra A, Ré C (2024). "Simple linear atten tion language models balance the recall-throughput tradeoff". arXiv. arXiv:2402.18668 [cs.CL].*

40. ^*Joshi M, Choi E, Weld D, Zettlemoyer L (2017). "triviaqa: A Large Scale Distantly Supervised Challenge Dataset for R eading Comprehension". arXiv e-prints. page arXiv:1705.03551. Available from: arXiv:1705.03551.*

41. ^*Novikova J, Dušek O, Rieser V (2017). "The E2E Dataset: New Challenges For End-to-End Generation". arXiv. arXiv: 1706.09254 [cs.CL].*

42. ^*Shimorina A, Gardent C (2018). "Handling Rare Items in Data-to-Text Generation". In: Proceedings of the 11th Int ernational Conference on Natural Language Generation. Tilburg University, The Netherlands: Association for Comp utational Linguistics. p. 360–370. Available from: http://aclweb.org/anthology/W18-6543.*

43. ^*Gardent C, Shimorina A, Narayan S, Perez-Beltrachini L. Creating training corpora for NLG micro-planners. In: Ba rzilay R, Kan M-Y, editors. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers. Association for Computational Linguistics; 2017. p. 179-188. doi:10.18653/v1/P17-1017.*

44. ^a, ^b*Biderman S, Schoelkopf H, Anthony Q, Bradley H, O'Brien K, Hallahan E, Khan MA, Purohit S, Prashanth USVSN, Raff E, Skowron A, Sutawika L, van der Wal O (2023). "Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling". arXiv. 2304.01373.*

45. ^a, ^b*Gemma Team, Mesnard T, Hardin C, Dadashi R, Bhupatiraju S, Pathak S, et al. Gemma: Open models based on ge mini research and technology. 2024. Available from: arXiv:2403.08295.*

46. ^*Together Computer (2023). "RedPajama: an Open Dataset for Training Large Language Models". Available from: h ttps://github.com/togethercomputer/RedPajama-Data.*

47. ^*Conover M, Hayes M, Mathur A, Xie J, Wan J, Shah S, Ghodsi A, Wendell P, Zaharia M, Xin R (2023). "Free Dolly: Int roducing the World's First Truly Open Instruction-Tuned LLM". https://www.databricks.com/blog/2023/04/12/dolly -first-open-commercially-viable-instruction-tuned-llm. Accessed 30 Jun 2023.*

48. ^a, ^b*Touvron H, Martin L, Stone K, Albert P, Almahairi A, Babaei Y, Bashlykov N, Batra S, Bhargava P, Bhosale S, Bikel D, Blecher L, Canton Ferrer C, Chen M, Cucurull G, Esiobu D, Fernandes J, Fu J, Fu W, Fuller B, Gao C, Goswami V, Goy al N, Hartshorn A, Hosseini S, Hou R, Inan H, Kardas M, Kerkez V, Khabsa M, Kloumann I, Korenev A, Koura PS, Lach aux MA, Lavril T, Lee J, Liskovich D, Lu Y, Mao Y, Martinet X, Mihaylov T, Mishra P, Molybog I, Nie Y, Poulton A, Reiz enstein J, Rungta R, Saladi K, Schelten A, Silva R, Smith EM, Subramanian R, Tan XE, Tang B, Taylor R, Williams A, Kuan JX, Xu P, Yan Z, Zarov I, Zhang Y, Fan A, Kambadur M, Narang S, Rodriguez A, Stojnic R, Edunov S, Scialom T. Llama 2: Open foundation and fine-tuned chat models, 2023. arXiv. 2023. Available from: https://arxiv.org/abs/230 7.09288.*

49. ^*Jiang AQ, Sablayrolles A, Mensch A, Bamford C, Chaplot DS, de las Casas D, Bressand F, Lengyel G, Lample G, Saulni er L, Lavaud LR, Lachaux MA, Stock P, Le Scao T, Lavril T, Wang T, Lacroix T, El Sayed W. Mistral 7B. 2023. Available from: arXiv [cs.CL]. arXiv:2310.06825.*

50. ^Zheng L, Chiang WL, Sheng Y, Zhuang S, Wu Z, Zhuang Y, Lin Z, Li Z, Li D, Xing EP, Zhang H, Gonzalez JE, Stoica I (2023). "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". arXiv. arXiv:2306.05685.
51. ^Daniele L, Suphavadeeprasit (2023). "Amplify-Instruct: Synthetically Generated Diverse Multi-turn Conversations for Effecient LLM Training." arXiv preprint arXiv:(comming soon).
52. ^Dubois Y, Galambosi B, Liang P, Hashimoto TB (2024). "Length-Controlled AlpacaEval: A Simple Way to Debias Au tomatic Evaluators". arXiv preprint arXiv:2404.04475.
53. ^Dubois Y, Li X, Taori R, Zhang T, Gulrajani I, Ba J, Guestrin C, Liang P, Hashimoto TB (2023). "AlpacaFarm: A Simul ation Framework for Methods that Learn from Human Feedback". arXiv. arXiv:2305.14387.
54. ^Cobbe K, Kosaraju V, Bavarian M, Chen M, Jun H, Kaiser L, Plappert M, Tworek J, Hilton J, Nakano R, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168. 2021.
55. ^Javaheripi M, Bubeck S (2023). "Phi-2: The surprising power of small language models". https://www.microsoft.co m/en-us/research/blog/phi-2-the-surprising-power-of-small-language-models/.
56. ^Azerbayev Z, Schoelkopf H, Paster K, Dos Santos M, McAleer S, Jiang AQ, Deng J, Biderman S, Welleck S (2023). "Lle mma: An Open Language Model For Mathematics". arXiv. arXiv:2310.10631 [cs.CL].
57. ^Wei J, Wang X, Schuurmans D, Bosma M, Xia F, Chi E, Le QV, Zhou D, et al. Chain-of-thought prompting elicits reas oning in large language models. Advances in neural information processing systems. 35:24824–24837, 2022.
58. a, bLu X, Liu Z, Liusie A, Raina V, Mudupalli V, Zhang Y, Beauchamp W (2024). "Blending Is All You Need: Cheaper, B etter Alternative to Trillion-Parameters LLM". arXiv. arXiv:2401.02994.
59. a, b, c, dGuha N, Chen M, Bhatia K, Mirhoseini A, Sala F, Ré C. "Embroid: Unsupervised prediction smoothing can imp rove few-shot classification." Advances in Neural Information Processing Systems. 36, 2024.
60. ^Penedo G, Malartic Q, Hesslow D, Cojocaru R, Cappelli A, Alobeidli H, Pannier B, Almazrouei E, Launay J (2023). "T he RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only". arXiv pr eprint arXiv:2306.01116. Available from: https://arxiv.org/abs/2306.01116.
61. a, bXiao S, Liu Z, Zhang P, Muennighoff N (2023). "C-Pack: Packaged Resources To Advance General Chinese Embed ding". arXiv. Available from: arXiv:2309.07597.
62. ^Varma P, Sala F, He A, Ratner A, Ré C (2019). "Learning Dependency Structures for Weak Supervision Models". arXi v. arXiv:1903.05844 [stat.ML].
63. ^Wang H, Polo FM, Sun Y, Kundu S, Xing E, Yurochkin M (2024). "Fusing Models with Complementary Expertise". ar Xiv. arXiv:2310.01542 [cs.LG].
64. ^Muqeeth M, Liu H, Liu Y, Raffel C (2024). "Learning to Route Among Specialized Experts for Zero-Shot Generalizat ion". arXiv. Available from: https://arxiv.org/abs/2402.05859.
65. ^Si C, Shi W, Zhao C, Zettlemoyer L, Boyd-Graber J (2023). "Getting MoRE out of Mixture of Language Model Reason ing Experts". arXiv. arXiv:2305.14628 [cs.CL].

66. ^Ding D, Mallick A, Wang C, Sim R, Mukherjee S, Ruhle V, Lakshmanan LVS, Awadallah AH (2024). "Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing". arXiv. arXiv:2404.14618 [cs.LG].
67. ^Hu QJ, Bieker J, Li X, Jiang N, Keigwin B, Ranganath G, Keutzer K, Upadhyay SK (2024). "RouterBench: A Benchmark for Multi-LLM Routing System". arXiv. **cs.LG**. Available from: https://arxiv.org/abs/2403.12031.
68. ^Chen L, Zaharia M, Zou J (2023). "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance". arXiv. arXiv:2305.05176.
69. ^Madaan A, Aggarwal P, Anand A, Potharaju SP, Mishra S, Zhou P, Gupta A, Rajagopal D, Kappaganthu K, Yang Y, Upadhyay S, Mausam, Faruqui M (2024). "AutoMix: Automatically Mixing Language Models". arXiv. arXiv:2310.12963 [cs.CL].
70. ^Ong I, Almahairi A, Wu V, Chiang WL, Wu T, Gonzalez JE, Kadous MW, Stoica I (2024). "RouteLLM: Learning to Route LLMs with Preference Data". arXiv. arXiv:2406.18665 [cs.LG].
71. ^Singla A, Singh A, Kukreja K (2023). "A bi-objective ϵ-constrained framework for quality-cost optimization in language model ensembles". arXiv. arXiv:2312.16119 [cs.LG].
72. ^Prabhakar R, Sivaramakrishnan R, Gandhi D, Du Y, Wang M, Song X, Zhang K, Gao T, Wang A, Li K, Sheng Y, Brot J, Sokolov D, Vivek A, Leung C, Sabnis A, Bai J, Zhao T, Gottscho M, Jackson D, Luttrell M, Shah MK, Chen E, Liang K, Jain S, Thakker U, Huang D, Jairath S, Brown KJ, Olukotun K (2024). "SambaNova SN40L: Scaling the AI Memory Wall with Dataflow and Composition of Experts". arXiv. **cs.AR**: 2405.07518.
73. ^Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G (2023). "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing". ACM Computing Surveys. **55** (9): 1–35.
74. ^Hou B, O'connor J, Andreas J, Chang S, Zhang Y. "Promptboosting: Black-box text classification with ten forward passes." In: International Conference on Machine Learning. PMLR; 2023. p. 13309–13324.
75. ^Zhang C, Liu L, Wang C, Sun X, Wang H, Wang J, Cai M (2024). "Prefer: Prompt ensemble learning via feedback-reflect-refine". Proceedings of the AAAI Conference on Artificial Intelligence. **38**: 19525–19532.
76. ^Peng X, Xing C, Choubey PK, Wu CS, Xiong C (2022). "Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning". arXiv preprint arXiv:2210.12587. arXiv:2210.12587.
77. ^Schick T, Schütze H (2020). "Exploiting cloze questions for few shot text classification and natural language inference". arXiv preprint arXiv:2001.07676. Available from: https://arxiv.org/abs/2001.07676.
78. ^Lester B, Al-Rfou R, Constant N (2021). "The power of scale for parameter-efficient prompt tuning". arXiv preprint arXiv:2104.08691. Available from: https://arxiv.org/abs/2104.08691.
79. ^Barzilay R, McKeown KR (2005). "Sentence Fusion for Multidocument News Summarization". Computational Linguistics. **31** (3): 297–328. doi:10.1162/089120105774321091. https://aclanthology.org/J05-3002.
80. ^Ravaut M, Joty S, Chen NF (2023). "Towards Summary Candidates Fusion". arXiv. arXiv:2210.08779 [cs.CL].
81. ^Lebanoff L, Dernoncourt F, Kim DS, Wang L, Chang W, Liu F (2020). "Learning to Fuse Sentences with Transformers for Summarization". arXiv. arXiv:2010.03726 [cs.CL].

82. ^Lebanoff L, Dernoncourt F, Kim DS, Chang W, Liu F (2020). "A Cascade Approach to Neural Abstractive Summarization with Content Selection and Fusion". arXiv. arXiv:2010.03722.

83. ^Huang Y, Feng X, Li B, Xiang Y, Wang H, Qin B, Liu T (2024). "Enabling Ensemble Learning for Heterogeneous Large Language Models with Deep Parallel Collaboration". arXiv. **cs.CL**: 2404.12715.

84. ^Mavromatis C, Karypis P, Karypis G (2024). "Pack of LLMs: Model Fusion at Test-Time via Perplexity Optimization". arXiv. Available from: https://arxiv.org/abs/2404.11531.

85. ^Wang J, Wang J, Athiwaratkun B, Zhang C, Zou J (2024). "Mixture-of-Agents Enhances Large Language Model Capabilities". arXiv. arXiv:2406.04692 [cs.CL].

86. ^Ilharco G, Ribeiro MT, Wortsman M, Gururangan S, Schmidt L, Hajishirzi H, Farhadi A (2023). "Editing Models with Task Arithmetic". arXiv. arXiv:2212.04089 [cs.LG].

87. ^Kumar S, Talukdar P (2021). "Reordering examples helps during priming-based few-shot learning". arXiv preprint arXiv:2106.01751.

88. a, bRubin O, Herzig J, Berant J (2021). "Learning to retrieve prompts for in-context learning". arXiv preprint arXiv:2112.08633. arXiv:2112.08633.

89. ^Do VT, Hoang VK, Nguyen DH, Sabahi S, Yang J, Hotta H, Nguyen MT, Le H (2024). "Automatic Prompt Selection for Large Language Models". arXiv preprint arXiv:2404.02717.

90. ^Liu J, Shen D, Zhang Y, Dolan B, Carin L, Chen W (2021). "What Makes Good In-Context Examples for GPT-3?" arXiv preprint arXiv:2101.06804.

91. ^Zhang Z, Zhang A, Li M, Smola A (2022). "Automatic chain of thought prompting in large language models". arXiv preprint arXiv:2210.03493. Available from: https://arxiv.org/abs/2210.03493.

92. ^Wu Z, Wang Y, Ye J, Kong L (2022). "Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering". arXiv preprint arXiv:2212.10375. arXiv:2212.10375.

93. ^Zhang Y, Feng S, Tan C (2022). "Active example selection for in-context learning". arXiv preprint arXiv:2211.04486. 2022.

94. ^Chang TY, Jia R (2022). "Data curation alone can stabilize in-context learning". arXiv preprint arXiv:2212.10378. **2022**.

95. a, bLiao C, Zheng Y, Yang Z (2022). "Zero-label prompt selection". arXiv preprint arXiv:2211.04668. Available from: https://arxiv.org/abs/2211.04668.

96. ^Lu Y, Bartolo M, Moore A, Riedel S, Stenetorp P (2021). "Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity". arXiv preprint arXiv:2104.08786.

97. ^Gonen H, Iyer S, Blevins T, Smith NA, Zettlemoyer L (2022). "Demystifying prompts in language models via perplexity estimation". arXiv preprint arXiv:2212.04037. arXiv:2212.04037.

98. ^Chen MF, Fu DY, Adila D, Zhang M, Sala F, Fatahalian K, Ré C. Shoring up the foundations: fusing model embeddings and weak supervision. In: Cussens J, Zhang K, editors. Proceedings of the Thirty-Eighth Conference on Uncertaint

y in Artificial Intelligence. PMLR; 2022. p. 357–367. Available from: https://proceedings.mlr.press/v180/chen22e.html.

99. a, b *Kobayashi H. "Frustratingly easy model ensemble for abstractive summarization." In: Riloff E, Chiang D, Hockenmaier J, Tsujii J, editors. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics; 2018 Oct–Nov. p. 4165–4176. doi:10.18653/v1/D18-1449. Available from: https://aclanthology.org/D18-1449.*
100. ^ *Bertsch A, Xie A, Neubig G, Gormley MR (2023). "It's MBR All the Way Down: Modern Generation Techniques Through the Lens of Minimum Bayes Risk". arXiv. arXiv:2310.01387 [cs.CL].*
101. ^ *Chai L, Sun H, Wang Z (2022). "An error consistency based approach to answer aggregation in open-ended crowdsourcing". Information Sciences. 608: 1029–1044. doi:10.1016/j.ins.2022.07.001. Link to article.*
102. ^ ***OpenAI**. "Learning to Reason with Large Language Models". https://openai.com/index/learning-to-reason-with-llms/, 2024. Accessed: 2024-10-29.*
103. ^ *Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E, Le Q, Zhou D (2023). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". arXiv. arXiv:2201.11903 [cs.CL].*
104. ^ *Deng Y, Zhang W, Chen Z, Gu Q (2024). "Rephrase and Respond: Let Large Language Models Ask Better Questions for Themselves". arXiv. arXiv:2311.04205 [cs.CL].*
105. ^ *Brown B, Juravsky J, Ehrlich R, Clark R, Le QV, Ré C, Mirhoseini A. Large Language Monkeys: Scaling Inference Compute with Repeated Sampling. arXiv [cs.LG]. 2024. Available from: https://arxiv.org/abs/2407.21787.*
106. ^ *Chen L, Davis JQ, Hanin B, Bailis P, Stoica I, Zaharia M, Zou J (2024). "Are More LLM Calls All You Need? Towards Scaling Laws of Compound Inference Systems". arXiv. arXiv:2403.02419 [cs.LG].*
107. ^ *Saad-Falcon J, Gamarra Lafuente A, Natarajan S, Maru N, Todorov H, Guha E, Buchanan EK, Chen M, Guha N, Ré C, Mirhoseini A. Archon: An architecture search framework for inference-time techniques, 2024. arXiv [cs.LG]. Available from: https://arxiv.org/abs/2409.15254.*

Declarations

Funding: We gratefully acknowledge the support of NIH under No. U54EB020405 (Mobilize), NSF under Nos. CCF2247015 (Hardware-Aware), CCF1763315 (Beyond Sparsity), CCF1563078 (Volume to Velocity), and 1937301 (RTML); US DEVCOM ARL under Nos. W911NF-23-2-0184 (Long-context) and W911NF-21-2-0251 (Interactive Human-AI Teaming); ONR under Nos. N000142312633 (Deep Signal Processing); Stanford HAI under No. 247183; NXP, Xilinx, LETI-CEA, Intel, IBM, Microsoft, NEC, Toshiba, TSMC, ARM, Hitachi, BASF, Accenture, Ericsson, Qualcomm, Analog Devices, Google Cloud, Salesforce, Total, the HAI-GCP Cloud Credits for Research program, the Stanford Data Science Initiative (SDSI), and members of the Stanford DAWN project: Meta, Google, and VMWare. NG is suported by the Stanford Interdisciplinary Graduate Fellowship (SIGF).

Potential competing interests: No potential competing interests to declare.