Efficient multi-prompt evaluation of LLMs

Felipe Maia Polo [1] **Ronald Xu** [2 3] **Lucas Weber** [4] **Mírian Silva** [2 5] **Onkar Bhardwaj** [2] **Leshem Choshen** [2 3]
Allysson Flavio Melo de Oliveira [2] **Yuekai Sun** [1] **Mikhail Yurochkin** [2 6]

Abstract

Most popular benchmarks for comparing LLMs rely on a limited set of prompt templates, which may not fully capture the LLMs' abilities and can affect the reproducibility of results on leaderboards. Many recent works empirically verify prompt sensitivity and advocate for changes in LLM evaluation. In this paper, we consider the problem of estimating the performance *distribution* across many prompt variants instead of finding a single prompt to evaluate with. We introduce PromptEval, a method for estimating performance across a large set of prompts borrowing strength across prompts and examples to produce accurate estimates under practical evaluation budgets. The resulting distribution can be used to obtain performance quantiles to construct various robust performance metrics (e.g., top 95% quantile or median). We prove that PromptEval consistently estimates the performance distribution and demonstrate its efficacy empirically on three prominent LLM benchmarks: MMLU, BIG-bench Hard, and LMentry. For example, PromptEval can accurately estimate performance quantiles across 100 prompt templates on MMLU with a budget equivalent to two single-prompt evaluations. Please check Maia Polo et al. [26] for the full paper.

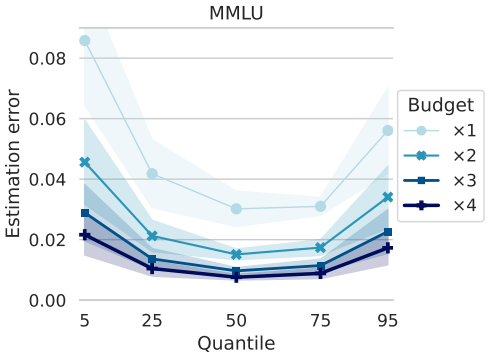

Figure 1: Average estimation error for performance quantiles across 100 templates given a limited budget (in multiples of one-template MMLU evaluations).

1. Introduction

In recent years, the rapid progress of large language models (LLMs) has significantly influenced various fields by enhancing automated text generation and comprehension. As these models advance in complexity and functionality, a key challenge that arises is their robust evaluation [30]. Common evaluation methods, which often rely on a single or limited number of prompt templates, may not adequately reflect the typical model's capabilities [47]. Furthermore, this approach can lead to unreliable and inconsistent rankings on LLM leaderboards, as different models may perform better or worse depending on the specific prompt template used. An ideal evaluation framework should minimize dependence on any single prompt template and instead provide a holistic summary of performance across a broad set of templates. Mizrahi et al. [28], for example, suggests using summary statistics, such as the average performance across many templates, as a way to compare the abilities of different LLMs. However, the main drawback of this method is the high computational cost when dealing with numerous templates and examples.

We introduce PromptEval, a method for efficient multi-prompt evaluation of LLMs. With a small number of evaluations, PromptEval estimates performance across different prompt templates. Our approach is grounded in robust theoretical foundations and utilizes well-established models from the fields of educational assessment and psychometrics, such as Item Response Theory (IRT) [6; 41; 5; 25]. Our method is based on a *parametric* IRT model that allows borrowing strength across examples and prompt templates to produce accurate estimates of all considered prompts with an evaluation budget comparable to evaluating a sin-

[*]Equal contribution [1]Department of Statistics, University of Michigan, USA [2]IBM Research [3]MIT [4]Department of Translation and Language Sciences, University of Pompeu Fabra, Spain [5]Department of Computer Science, Federal University of Minas Gerais, Brazil [6]MIT-IBM Watson AI Lab. Correspondence to: Felipe Maia Polo <felipemaiapolo@gmail.com>.

Efficient Systems for Foundation Models Workshop, International Conference on Machine Learning (ICML), Vienna, Austria. 2024.

gle prompt. In Figure 1, we demonstrate the ability of our method to jointly estimate various performance quantiles across 100 prompt templates with evaluation budget ranging from one to four times of a conventional single-prompt evaluation on MMLU [16].

Performance distribution across prompts can be used to accommodate various contexts when comparing LLMs [8]. For example, it can be used to compute mean as suggested by Mizrahi et al. [28]. One can also use performance distributions directly to compare LLMs via various notions of stochastic dominance for risk-sensitive scenarios [29]. Here we primarily focus on the full distribution and its quantiles as they provide a flexible statistic that can inform decisions in varying contexts. For instance, a typical model performance corresponds to a median (50% quantile), 95% quantile can be interpreted as performance achievable by an expert prompt engineer, while 5% quantile is of interest in consumer-facing applications to quantify low-end performance for a user not familiar with prompt engineering. We also demonstrate (Appendix H) how our method can be used to improve best prompt identification [37]. Our main contributions are:

- We propose (§3) a novel method called PromptEval which permits efficient multi-prompt evaluation of LLMs across different prompt templates with a limited number of evaluations.
- We theoretically show (§I) that PromptEval has desirable statistical properties such as consistency in estimating performance distribution and its quantiles.
- We practically demonstrate (§4) efficacy of PromptEval in estimating performance across 100+ prompts and finding the best-performing prompt for various LLMs using data derived from three popular benchmarks: MMLU [16], BIG-bench Hard (BBH) [40], and LMentry [11].
- We conduct the first large-scale study of prompt sensitivity of 15 popular open-source LLMs on MMLU. We present our findings based on evaluating 100 prompt templates in Appendix K and will release the evaluation data.

In Appendix A, we talk in detail about related work.

2. Problem statement

In this section, we describe the setup we work on and what our objectives are. Consider that we want to evaluate a large language model (LLM) in a certain dataset composed of J examples (also known as questions or items in the literature) and each one of the examples is responded to by the LLM through prompting; we assume that there exists I different prompt templates that can be used to evaluate the LLM. After the prompt template $i \in \mathcal{I} \triangleq [I]$ and example $j \in \mathcal{J} \triangleq [J]$ are channeled through the LLM, some grading system generates a correctness score $Y_{ij} \in \{0, 1\}$, which assumes 1 when the prompt template i and example j jointly yield a correct response and 0 otherwise[1]. For each one of the prompt templates $i \in \mathcal{I}$, we can define its performance score as $S_i \triangleq \frac{1}{J} \sum_{j \in \mathcal{J}} Y_{ij}$. The performance scores S_i's can have a big variability, making the LLM evaluation reliant on the prompt choice. To have a comprehensive evaluation of the LLM, we propose computing the full *distribution of performances* and its corresponding quantile function, *i.e.*,

$$F(x) \triangleq \frac{1}{I} \sum_{i \in \mathcal{I}} \mathbb{1}_{[S_i, \infty)}(x) \quad \text{and} \quad Q(p) \triangleq \inf\{x \in \mathbb{R} : F(x) \geq p\}. \tag{2.1}$$

The main challenge in obtaining this distribution is that it can be very expensive since the exact values for the performance scores S_i's require $I \cdot J$ evaluations. In this paper, we assume that only a small fraction of evaluations is available, *e.g.*, $< 5\%$ of the total number of possible $I \cdot J$ evaluations, but we still aim to accurately estimate the performance distribution and its quantiles. More concretely, we assume the correctness scores Y_{ij}'s are only evaluated for a small set of indices $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{J}$; in compact notation, we define $Y_{\mathcal{E}} \triangleq \{Y_{ij}\}_{(i,j) \in \mathcal{E}}$. Here, the letter $\mathcal{E}$ stands for *evaluations*. Using the observed data $Y_{\mathcal{E}}$, our main objective is to estimate the performance scores distribution F (resp. quantile function Q), *i.e.*, computing a function $\hat{F}$ (resp. $\hat{Q}$) that is *close* to F (resp. Q).

3. Performance distribution and quantiles estimation

3.1. The correctness model

We assume the observations Y_{ij}'s are independently sampled from a Bernoulli model parameterized by prompt/example-specific parameters. That is, we assume

$$Y_{ij} \sim \text{Bernoulli}(\mu_{ij}), \tag{3.1}$$

where μ_{ij} denotes the mean of the Bernoulli distribution specific to prompt format i and example j. We can write $\mu_{ij} = \mu(\theta_i, \beta_j)$, where θ_i's are prompt-specific parameters, β_j's are example-specific parameters and μ is a function that maps those parameters to the Bernoulli mean. This probabilistic model is very general and comprehends factor models such as the large class of Item Response Theory (IRT) models [6; 41; 5; 25]; as we will see, our model can be seen as a general version of an IRT model. For generality purposes, we assume that the parameters θ_i's and β_j's can be written as functions of prompt-specific (x_i's)

[1] In some cases, the correctness score may be a bounded number instead of binary – see Appendix D.

and example-specific (z_j's) vectors of covariates. That is, we assume $\theta_i = f_\psi(x_i)$ or $\beta_j = g_\gamma(z_j)$, where ψ and γ are global parameters that can be estimated. These covariates can be, for example, embeddings of prompt templates in the case of x_i's and some categorization or content of each of the examples in the case of z_j's. In this work, we adopt $\mu(\theta_i, \beta_j) = \sigma(\theta_i - \beta_j) = \sigma(f_\psi(x_i) - g_\gamma(z_j))$, where σ denotes the standard logistic function and the functions f_ψ and g_γ have their image in $\mathbb{R}$. That is, our model assumes that

$$\mathbb{P}(Y_{ij} = 1; \psi, \gamma) = \frac{1}{1 + \exp[-(f_\psi(x_i) - g_\gamma(z_j))]}. \quad (3.2)$$

The functions f_ψ and g_γ can be represented with neural networks. On the simpler side, one could just assume f_ψ and g_γ are linear, that is, $\theta_i = \psi^\top x_i$ or $\beta_j = \gamma^\top z_j$. We consider that, in some cases, a constant can be embedded in x_i in order to include an intercept in the model. When x_i and z_j are one-hot encoded vectors, *i.e.*, vector of zeros but with 1's on the entries i and j, the model in (3.2) reverts to a popular IRT model known as the Rasch model [15; 7], which is widely used in fields such as recommendation systems [39] and educational testing [9]. One major limitation of the basic Rasch model is that the number of parameters is large, compromising the quality of the estimates for ψ and γ when either the number of prompt formats I or the number of examples J is large and $|\mathcal{E}|$ is small, *i.e.*, only a few evaluations are carried out. This degradation in the quality of the estimates can directly affect the quality of the performance distribution estimates. Finally, we fit the parameters ψ and γ, obtaining the estimates $\hat{\psi}$ and $\hat{\gamma}$, by maximizing the log-likelihood of the observed data (negative cross-entropy loss), *i.e.*,

$$(\hat{\psi}, \hat{\gamma}) \in \arg\max_{\psi,\gamma} \sum_{(i,j)\in\mathcal{E}} Y_{ij} \log \mathbb{P}(Y_{ij} = 1; \psi, \gamma) + (1 - Y_{ij}) \log\left(1 - \mathbb{P}(Y_{ij} = 1; \psi, \gamma)\right). \quad (3.3)$$

Realize that fitting the model with linear/affine f_ψ and g_γ, including the Rasch model case[2], reduces to fitting a logistic regression model with x_i and z_j as the covariates. In the experiments Section 4, we explore some different options of covariates for both templates and examples.

3.2. Performance distribution and quantiles estimation using the correctness model

The model in (3.1) can be naturally used for performance estimation. That is, after observing $Y_\mathcal{E}$, the best approximation (in the mean-squared-error sense) for the performance of prompt format $i \in \mathcal{I}$, S_i, is given by the following conditional expectation

$$\mathbb{E}[S_i \mid Y_\mathcal{E}] = \frac{\lambda_i}{|\mathcal{J}_i|} \sum_{j\in\mathcal{J}_i} Y_{ij} + \frac{1-\lambda_i}{|\mathcal{J} \setminus \mathcal{J}_i|} \sum_{j\notin\mathcal{J}_i} \mu_{ij}$$

where $\mathcal{J}_i \triangleq \{j \in \mathcal{J} : (i,j) \in \mathcal{E}\}$ and $\lambda_i = |\mathcal{J}_i|/J$. In practice, computing $\mathbb{E}[S_i \mid Y_\mathcal{E}]$ is impossible because the parameters θ_i's and β_j's are unknown. We can, however, use a plug-in estimator for the conditional expectation using their maximum likelihood estimators, changing μ_{ij} for $\sigma\big(f_{\hat{\psi}}(x_i) - g_{\hat{\gamma}}(z_j)\big)$. The basic version of this estimator, when no elaborate covariates (*e.g.*, embeddings) are included, is known as the Performance-IRT (pIRT) estimator [31]. We can apply our extended version of pIRT, which we call X-pIRT, to estimate the performance distribution across prompt templates. After observing $Y_\mathcal{E}$ and fitting $(\hat{\psi}, \hat{\gamma})$, we can compute $\hat{S}_i \triangleq \hat{\mathbb{E}}[S_i \mid Y_\mathcal{E}]$ for all $i \in \mathcal{I}$. Then, we define our estimators for the distribution of performances and its corresponding quantile function 2.1 as $\hat{F}$ and $\hat{Q}$ by substituting S_i by $\hat{S}_i$ in (2.1). We name the procedure of obtaining $\hat{F}$ and $\hat{Q}$ as PromptEval and summarize it in Algorithm 1. We include details on how to sample $\mathcal{E}$ in Appendix B.

Algorithm 1: PromptEval

Input: (i) $Y_\mathcal{E}$, (ii) covariates x_i's and z_j's.

Output: Estimates for the performances distribution and its quantile function (2.1).

Fit ψ and γ using observed scores $Y_\mathcal{E}$ and covariates x_i's and z_j's using maximum likelihood (3.3).

For each $i \in \mathcal{I}$, compute $\hat{S}_i = \hat{\mathbb{E}}[S_i \mid Y_\mathcal{E}]$ using X-pIRT.

Compute estimates $\hat{F}(\cdot) \triangleq \frac{1}{I}\sum_{i\in\mathcal{I}} \mathbb{1}_{[\hat{S}_i,\infty)}(\cdot)$ and $\hat{Q}(\cdot) \triangleq \inf\{x \in \mathbb{R} : \hat{F}(x) \geq \cdot\}$.

return $\hat{F}$ and $\hat{Q}$.

4. Assessing multi-prompt evaluation strategies

General assessment We assess the performance distribution and quantile function estimation methodology introduced in §3 in estimating the performance of LLMs and different prompt formats on data from three popular benchmarks. For a given LLM and a dataset, we consider two evaluation steps. First, we estimate some quantiles of interest (*e.g.*, $5/25/50/75/95$-th) for the performance distribution across prompt formats and compare them with the true quantiles, that is, for some $p \in [0,1]$, we use $|Q(p) - \hat{Q}(p)|$ to measure the quality of our estimations. Second, we compare the full performance distribution with the estimated distribution, *i.e.*, in this case, all quantiles are considered. To compare the full performance distribution F and its estimate $\hat{F}$, both defined in §3, we use the Wasserstein 1-distance

[2] For a detailed fitting procedure in the Rasch model case, please check Chen et al. [7].

which is equivalent to the average quantile estimation error in this case, *i.e.*, $\frac{1}{I}\sum_{i=1}^{I}|S_{(i)}-\hat{S}_{(i)}|$, where $S_{(i)}$ (resp. $\hat{S}_{(i)}$) is the i-th smallest value in $\{S_i\}_{i\in\mathcal{I}}$ (resp. $\{\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]\}_{i\in\mathcal{I}}$). The second analysis is presented in Appendix F

Data We use data derived from three popular benchmarks: MMLU [16], BIG-bench Hard (BBH) [40], and LMentry [11]. In the following, we give more details about each one of the used datasets.

- MMLU is a multiple choice QA benchmark consisting of 57 subjects (tasks) comprising approximately 14k examples. We ran 15 different open-source LLMs (including different versions of Llama-3 [27], Mistral [17], and Gemma [14]) combined with 100 different prompt variations for each one of the MMLU tasks. We found that, within each one of the MMLU tasks, prompt templates can have great variability in their performances, making within-task analysis most suitable for assessing our method. More details and analysis of the collected data can be found in Appendices J and K.
- BIG-bench Hard (BBH) is a curated subset of BIG-bench [38], containing challenging tasks on which LLMs underperform the average human score. For BBH, we use the evaluation scores released by [28]. The evaluation data includes 11 open-source LLMs combined with a different number of prompt variations, ranging from 136 to 188 formats, for 15 tasks containing 100 examples each.
- LMentry consists of simple linguistic tasks designed to capture explainable and controllable linguistic phenomena. Like BBH, we use data generated by [28]. The authors made available the full evaluation data from 16 open-source LLMs combined with a different number of prompt variations, ranging from 226 to 259 formats, for 10 tasks containing from 26 to 100 examples each.

Methods and baselines We consider different variations of the model presented in (3.2) coupled with Algorithm 1; for all variations, we use linear f_ψ and g_γ. The most basic version of the model in (3.2) assumes x_i and z_j are one-hot encoded vectors, *i.e.*, vector of zeros with 1's on the entries i and j, reverting the model to a Rasch model [15; 7]. Despite its simplicity, we show that it can perform well in some cases. A more advanced instance of (3.2) assumes x_i are either obtained using a sentence transformer [33] to embed prompt templates or by extracting discrete covariates from the text, *e.g.*, as the presence of line breaks, colons *etc.*(see Appendix Table 2). An example of a prompt template for LMentry used by Mizrahi et al. [28] is "*Can {category} be used to classify all the {words} provided? Respond with either "yes" or "no"*." Our method also allows using example covariates z_j, however, upon preliminary tests with sentence transformer we didn't observe improvements and chose to use one-hot-encoded vectors as in the basic Rasch model to represent examples. Next, we detail the methods for obtaining the prompt covariates:

- *Prompt embeddings*. We embed prompt templates using a pre-trained sentence transformer variant [18] and reduce their dimensionality to $d = 25$ using PCA. This is the most general solution that also works well in practice. We call it EmbPT.
- *Fine-tuned prompt embeddings*. Sentence transformers in general might not be most suitable for embedding prompt templates, thus we also consider fine-tuning BERT [10] as an embedder. To do so, we use evaluation data for all examples and prompt formats from a subset of LLMs (these LLMs are excluded when assessing the quality of our estimators) and fine-tune `bert-base-uncased` to predict Y_{ij} as in (3.3). We call this variation EmbFT and provide additional details in Appendix L. We acknowledge that obtaining such evaluation data for fine-tuning might be expensive, however, it might be justified in some applications if these embeddings provide sufficient savings for future LLM evaluations.
- *Discrete prompt covariates*. For BBH and LMentry, we coded a heuristic function that captures frequently occurring differences in common prompting templates. Examples of such covariates are the number of line breaks or the count of certain special characters (*e.g.*, dashes or colons). Each one of these covariates is encoded in x_i for each one of the prompt templates $i \in \mathcal{I}$. A full list of the used heuristics is detailed in Appendix M. For MMLU, we adopted approach of [36] to generate prompt variations via templates (see Algorithm 3), which also provides a natural way to construct the covariates, *e.g.*, the presence of dashes or colons.

To the best of our knowledge, the methods introduced in §3 are the first ones handling the problem of efficient evaluation of performance *distribution* of LLMs across multiple prompts. Thus, we compare different variations of our method with one natural baseline ("avg") which estimates S_i by simply averaging Y_{ij}, that is, using the estimates $\hat{S}_i^{\text{avg}} = \frac{1}{|\mathcal{J}_i|}\sum_{j\in\mathcal{J}_i} Y_{ij}$. The estimates for the distribution and quantile function are then obtained by computing $\hat{Q}$ and $\hat{F}$ using $\hat{S}_i^{\text{avg}}$ instead of $\hat{S}_i$. To make comparisons fair, we sample the data using Algorithm 2 for all methods and the baseline.

Key results We investigate the effectiveness of the different variations of PromptEval (PE) against the "avg" baseline strategy in quantile estimation and overall performance distribution estimation across prompt templates. In total,

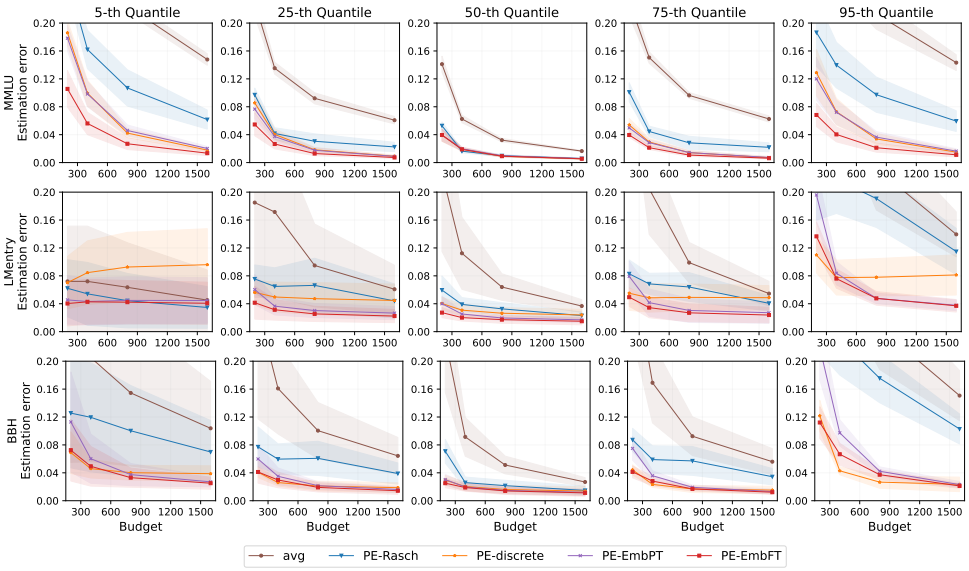

Figure 2: Performance quantile estimation errors for varying quantiles (columns) and benchmarks (rows).

we consider five variations of PromptEval: (i) PE-Rasch (model in (3.2) is a Rach model), (ii) PE-discrete (discrete covariates are used for prompt templates), (iii) PE-EmbPT (pre-trained LLM embeddings are used for prompt templates), and (iv) PE-EmbFT (fine-tuned LLM embeddings are used for prompt templates). Within each one of the benchmarks, we conduct a different experiment for each one of the tasks, LLMs, and 5 random seeds used when sampling $Y_{\mathcal{E}}$. We report the average estimation error across tasks, LLMs, and seeds, while the error bars are for the average estimation errors across LLMs. We collect results for four different numbers of total evaluations, where $|\mathcal{E}| \in \{200, 400, 800, 1600\}$. To make our results more tangible, 200 evaluations are equivalent, on average, to 1.15% to the total number of evaluations on BBH, 0.88% to the total number of evaluations on LMentry, and 0.81% to the total number of evaluations on MMLU.

Our results for quantile estimation are presented in Figure 2. As before, even the simplest version of our method (*PE-Rasch*) does much better than the considered baseline. For all the other variations of PromptEval, estimating extreme quantiles is usually hard and needs more evaluations, while more central ones (*e.g.*, median) can be accurately estimated with 200 evaluations, providing more than 100x compute saving in most cases. Regarding the different variations of PromptEval, we found that the pre-trained embeddings are robust across benchmarks, while the discrete covariates could not do well on LMentry data. Using covariates obtained via fine-tuning the BERT model provides some further improvements, for example, for extreme quantiles and small evaluation budget settings on MMLU. However, fine-tuning requires collecting large amounts of evaluation data and in most cases, we anticipate that it would be more practical to use PromptEval with pre-trained embedder and moderate evaluation budget instead.

5. Conclusion

PromptEval enables a more comprehensive evaluation of LLMs. We hope that comparing distributions or quantiles across many prompt variants will enable more robust leaderboards and address the common concern of comparing LLMs with a single pre-defined prompt. Prior to our work, a major limitation of such evaluation was its cost. We demonstrated empirically across several popular benchmarks that our method can produce accurate performance distribution and quantile estimates at the cost of 2-4 single-prompt evaluations, out of hundreds possible. However, several questions remain: how to decide on the set of prompts for evaluation and how to best utilize our distribution estimates for comparison in various contexts. For the former, we utilized suggestions from prior work [28; 36] and for the latter, we primarily focused on quantiles as well-established robust performance measures.

Besides evaluation, another common problem in practice is finding the best prompt for a given task. Our method can be applied in this setting when there is a pre-defined set of candidate prompts (Figure 7). However, several recent works [32; 48; 23; 49] demonstrate the benefits of dynamically generating new prompt candidates. For example, Prasad et al. [32] propose an evolutionary algorithm that creates new prompts based on the ones that performed well at an earlier iteration. Extending PromptEval to accommodate an evolving set of prompt candidates is an interesting future work direction.

References

[1] Shir Ashury-Tahan, Benjamin Sznajder, Leshem Choshen, Liat Ein-Dor, Eyal Shnarch, and Ariel Gera. Label-efficient model selection for text generation. *arXiv preprint arXiv:2402.07891*, 2024.

[2] Mohammad Javad Azizi, Branislav Kveton, and Mohammad Ghavamzadeh. Fixed-budget best-arm identification in structured bandits. *arXiv preprint arXiv:2106.04763*, 2021.

[3] Elron Bandel, Yotam Perlitz, Elad Venezian, Roni Friedman-Melamed, Ofir Arviv, Matan Orbach, Shachar Don-Yehyia, Dafna Sheinwald, Ariel Gera, Leshem Choshen, Michal Shmueli-Scheuer, and Yoav Katz. Unitxt: Flexible, shareable and reusable data preparation and evaluation for generative ai, 2024.

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

[5] Justyna Brzezińska. Item response theory models in the measurement theory. *Communications in Statistics-Simulation and Computation*, 49(12):3299–3313, 2020.

[6] Li Cai, Kilchan Choi, Mark Hansen, and Lauren Harrell. Item response theory. *Annual Review of Statistics and Its Application*, 3:297–321, 2016.

[7] Yunxiao Chen, Chengcheng Li, Jing Ouyang, and Gongjun Xu. Statistical inference for noisy incomplete binary matrix. *Journal of Machine Learning Research*, 24(95):1–66, 2023.

[8] Leshem Choshen, Ariel Gera, Yotam Perlitz, Michal Shmueli-Scheuer, and Gabriel Stanovsky. Navigating the modern evaluation landscape: Considerations in benchmarks and frameworks for large language models (llms). In *International Conference on Language Resources and Evaluation*, 2024. URL https://api.semanticscholar.org/CorpusID:269804253.

[9] Douglas H Clements, Julie H Sarama, and Xiufeng H Liu. Development of a measure of early mathematics achievement using the rasch model: The research-based early maths assessment. *Educational Psychology*, 28(4):457–482, 2008.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[11] Avia Efrat, Or Honovich, and Omer Levy. Lmentry: A language model benchmark of elementary language tasks. *arXiv preprint arXiv:2211.02069*, 2022.

[12] Ludwig Fahrmeir and Heinz Kaufmann. Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models. *The Annals of Statistics*, 13(1):342–368, 1985.

[13] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

[14] Team Gemma, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[15] Rasch Georg. Probabilistic models for some intelligence and attainment tests. *Copenhagen: Institute of Education Research*, 1960.

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

[17] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[18] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for

Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

[19] Maurice G Kendall and B Babington Smith. The problem of m rankings. *The annals of mathematical statistics*, 10(3):275–287, 1939.

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

[21] John P Lalor, Hao Wu, and Hong Yu. Building an evaluation scale using item response theory. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2016, page 648. NIH Public Access, 2016.

[22] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023.

[23] Yujian Betterest Li and Kai Wu. Spell: Semantic prompt evolution based on a llm. *arXiv preprint arXiv:2310.01260*, 2023.

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

[25] FM Lord, MR Novick, and Allan Birnbaum. Statistical theories of mental test scores. 1968.

[26] Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of llms. *arXiv preprint arXiv:2405.17202*, 2024.

[27] Meta. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3, 2024.

[28] Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. State of what art? a call for multi-prompt llm evaluation. *arXiv preprint arXiv:2401.00595*, 2023.

[29] Apoorva Nitsure, Youssef Mroueh, Mattia Rigotti, Kristjan Greenewald, Brian Belgodere, Mikhail Yurochkin, Jiri Navratil, Igor Melnyk, and Jerret Ross. Risk assessment and statistical significance in the age of foundation models. *arXiv preprint arXiv:2310.07132*, 2023.

[30] Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. Efficient benchmarking (of language models). *arXiv preprint arXiv:2308.11696*, 2023.

[31] Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024.

[32] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3827–3846, 2023.

[33] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[34] Sidney Resnick. *A probability path*. Springer, 2019.

[35] Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4486–4503, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.346. URL https://aclanthology.org/2021.acl-long.346.

[36] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*, 2023.

[37] Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. Best arm identification for prompt learning under a limited budget. *arXiv preprint arXiv:2402.09723*, 2024.

[38] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià

Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[39] Alain Starke, Martijn Willemsen, and Chris Snijders. Effective user interface designs to increase energy-efficient behavior in a rasch-based energy recommender system. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 65–73, 2017.

[40] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[41] Wim J Van der Linden. *Handbook of item response theory: Three volume set*. CRC Press, 2018.

[42] Clara Vania, Phu Mon Htut, William Huang, Dhara Mungra, Richard Yuanzhe Pang, Jason Phang, Haokun Liu, Kyunghyun Cho, and Samuel R Bowman. Comparing test sets with item response theory. *arXiv preprint arXiv:2106.00840*, 2021.

[43] Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. Anchor points: Benchmarking models with much fewer examples. *arXiv preprint arXiv:2309.08638*, 2023.

[44] Anton Voronov, Lena Wolf, and Max Ryabinin. Mind your format: Towards consistent evaluation of in-context learning improvements. *arXiv preprint arXiv:2401.06766*, 2024.

[45] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.

[46] Lucas Weber, Elia Bruni, and Dieuwke Hupkes. The icl consistency test. *arXiv preprint arXiv:2312.04945*, 2023.

[47] Lucas Weber, Elia Bruni, and Dieuwke Hupkes. Mind the instructions: a holistic evaluation of consistency and interactions in prompt-based learning. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pages 294–313, 2023.

[48] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[49] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*, 2023.

[50] Qinyuan Ye, Harvey Yiyun Fu, Xiang Ren, and Robin Jia. How predictable are large language model capabilities? a case study on big-bench. *arXiv preprint arXiv:2305.14947*, 2023.

A. Related work

LLMs' sensitivity to prompt templates The sensitivity of Large Language Models (LLMs) to the prompts is well-documented. For example, Sclar et al. [36] revealed that subtle variations in prompt templates in few-shot settings can lead to significant performance discrepancies among several open-source LLMs, with differences as large as 76 accuracy points in tasks from the SuperNaturalInstructions dataset [45]. Additionally, they report that the performance of different prompt templates tends to correlate weakly between models. This finding challenges the reliability of evaluation methods that depend on a single prompt template. To measure LLMs sensitivity, the researchers suggested calculating a "performance spread," which represents the difference between the best and worst performances observed. Mizrahi et al. [28] conducted a complementary analysis using state-of-the-art models and subsets of BigBench and LMentry [38; 11]. The authors arrive at similar conclusions with respect to LLMs' sensitivity to the used prompt templates and empirically showed that the LLM ranking considering different formats are usually weakly or intermediately correlated with each other. As a solution to the lack of robustness in LLM evaluation, the authors propose the use of summary statistics, as the average performance, for LLM evaluation. Some other works, *e.g.*, Voronov et al. [44]; Weber et al. [47; 46], show that even when in-context examples are given to the models, the prompt templates can have a big impact on the final numbers, sometimes reducing the performance of the strongest model in their analyses to a random guess level [44]. In a different direction, Shi et al. [37] acknowledges that different prompt templates have different performances and proposes using best-arm-identification to efficiently select the best template for an application at hand. One major bottleneck is still on how to efficiently compute the performance distribution for LLMs over many prompt templates; we tackle this problem.

Efficient evaluation of LLMs The escalating size of models and datasets has led to increased evaluation costs. To streamline evaluations, Ye et al. [50] considered minimizing the number of *tasks* within Big-bench [38]. Additionally, Perlitz et al. [30] observed that evaluations on HELM [24] rely on diversity across datasets, though the quantity of examples currently utilized is unnecessarily large. Perlitz et al. [30] also highlighted the problems in evaluating with insufficient prompts and called to evaluate on more, suggesting evaluating the typical behavior by sampling prompts and examples together by employing stratified sampling, where subscenarios give the strata; in our work, we also apply stratification but consider prompt templates and examples to give the strata. To accelerate evaluations for classification tasks, Vivek et al. [43] suggested clustering evaluation examples based on model confidence in the correct class. More recently, Polo et al. [31] empirically showed that it is possible to shrink the size of modern LLM benchmarks and still retain good estimates for LLMs' performances. Similarly (and in parallel to this work) Ashury-Tahan et al. [1] recognized unlabeled examples that better distinguish between models or prompts, by analyzing model outputs on them, hence saving costly annotation for them. Despite these advancements in streamlining LLM evaluations, there are no other works that propose a general and efficient method to estimate the benchmark performance of LLMs across prompt templates to the best of our knowledge.

Item response theory (IRT) IRT [6; 41; 5; 25] is a collection of statistical models initially developed in psychometrics to assess individuals' latent abilities through standardized tests but with increasing importance in the fields of artificial intelligence and natural language processing (NLP). For example, Lalor et al. [21] used IRT's latent variables to measure language model abilities, Vania et al. [42] applied IRT to benchmark language models and examine the saturation of benchmarks, and Rodriguez et al. [35] explored various uses of IRT with language models, including predicting responses to unseen items, categorizing items by difficulty, and ranking models. Recently, Polo et al. [31] suggested using IRT for efficient LLM performance evaluation, introducing the Performance-IRT (pIRT) estimator to evaluate LLMs. Our quantile estimation methodology is built upon pIRT.

B. Sampling $Y_{\mathcal{E}}$

We have assumed $Y_{\mathcal{E}}$ is given so far. In practice, however, we need to choose $\mathcal{E}$, with $|\mathcal{E}| \leq B$ where $B \in \mathbb{N}$ is the budget, and then sample the entries Y_{ij} for all $(i, j) \in \mathcal{E}$. One possible option is sampling (i, j) without replacement from $\mathcal{I} \times \mathcal{J}$ giving the same sampling probability to all entries. This option is, however, suboptimal because of its high instability: with a high chance, there will be some prompt formats (or examples) with a very low number of evaluations while others will have many. A more stable solution is given by Algorithm 2, which balances the number of times each prompt format and examples are evaluated. Algorithm 2 can be seen as two-way stratified random sampling in which the number of examples observed for each prompt format is (roughly) the same and the number of prompt formats that observe each one of the examples is (roughly) the same.

Algorithm 2: Two-way balanced sampling

Input: (i) sets $\mathcal{I}$ and $\mathcal{J}$, (ii) budget B.
Output: Observed indices $\mathcal{E}$.
Initialize $\mathcal{E} = \{\}$.
for $b = 0$ *to* $B - 1$ **do**
  Among $i \in \mathcal{I}$ with the least number of evaluations, randomly pick one of them and call it $\hat{i}$.
  Among $j \in \mathcal{J}$ such that $(\hat{i}, j) \notin \mathcal{E}$, randomly pick $\hat{j}$ from the ones with the least number of evaluations.
  Update $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\hat{i}, \hat{j})\}$
return $\mathcal{E}$.

C. Limitations

While our method provides a more reliable measure and a more flexible one, it assumes multiple prompts. Thus, if the question of which single prompt should be used was a challenge for old benchmarks, which set of prompt templates to use is the challenge now. While methods have been suggested for generating multiple prompts and diversifying those [28], and while when many prompts are suggested the choice of each one is likely less critical, it is a limitation future work should consider.

Another limitation we note is that we do not focus on prompt engineering and do not solve this problem. While this would have been a crucial contribution for the field, we assume a set of prompts and assume we only care about evaluation and not training, which make for a useful and common setting, but it does not encompass this goal.

D. Adapting the correctness model for bounded Y_{ij}

There might be situations in LLM evaluation in which $Y_{ij} \notin \{0, 1\}$ but $Y_{ij} \in [0, 1]$. For example, in AlpacaEval 2.0 [22], the response variable is bounded and can be translated to the interval $[0, 1]$. Also, some scenarios of HELM [24] and the Open LLM Leaderboard [4] have scores in $[0, 1]$. One possible fix is changing the model for Y_{ij}. For example, if Y_{ij} are continuous, the Beta model would be appropriate. Another possibility that offers a more immediate fix is binarizing Y_{ij} as proposed by Polo et al. [31]. That is, using a training set containing correctness data from L LLMs, we could find a constant c such that $\sum_{i,j,l} Y_{ijl} \approx \sum_{i,j,l} \mathbb{1}[Y_{ijl} \geq c]$, where the index l represents each LLM in the training set. Then, we define $\tilde{Y}_{ij} \triangleq \mathbb{1}[Y_{ij} \geq c]$ and work with this newly created variable.

E. Computing resources

All experiments were conducted using a virtual machine with 32 cores. The results for each benchmark separately can be obtained within 3-6 hours.

For fine-tuning BERT embeddings, we employ multiple NVIDIA A30 GPUs with 24 GB vRAM, requiring 70 hours of training and an additional approximately 350 hours for hyperparameter search. Fine-tuning can be conducted on GPUs with smaller capacities.

F. Distribution estimation results

Our results for distribution estimation can be seen in Figure 3. We see that, in general, all variations of PromptEval, including its simplest version (*PE-Rasch*), can do much better in distribution estimation when compared to the baseline. Among our methods, the ones that use covariates are the best ones.

G. Estimation errors by task

In Figures 4, 5, and 6, we analyze the Wasserstein 1-distance per task for each benchmark when using the method PE-EmbPT, a robust and versatile variation of PromptEval. The results show that for BBH and LMentry, the estimation errors (Wasserstein 1-distance) are more uniform across tasks compared to MMLU, where some tasks exhibit higher estimation

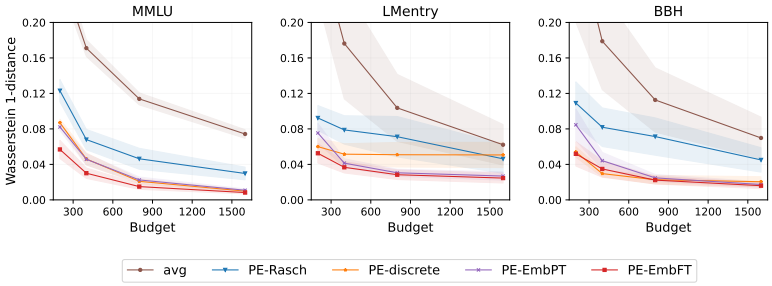

Figure 3: Performance distribution estimation errors measured with Wasserstein-1 distance on three benchmarks.

errors. This discrepancy occurs because all tasks in BBH and LMentry have the same number of examples, whereas tasks in MMLU, particularly those with higher estimation errors, have a significantly larger number of examples when compared to the others. In those cases, a larger number of evaluations is recommended.

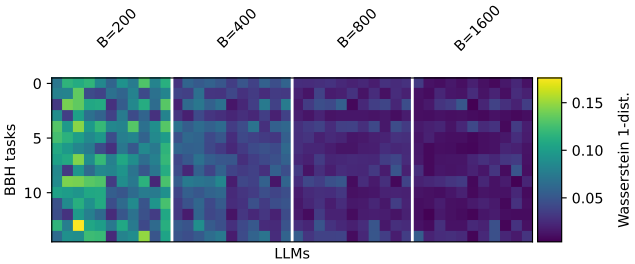

Figure 4: Estimation error for the BBH tasks.

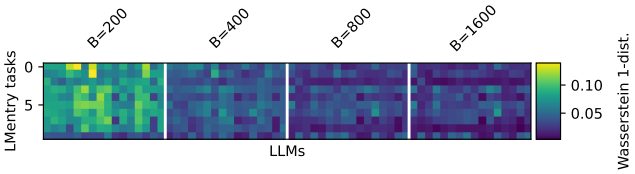

Figure 5: Estimation error for the LMentry tasks.

H. Best-prompt identification

The best-prompt identification task [37] is to find the best prompt from a set of fixed templates, *i.e.*, the one that gives the best performance for a task at hand. Shi et al. [37] propose framing this problem as a bandit problem and using a linear model or an MLP to predict the performance of each prompt template. To apply PromptEval in this setting we use our model (3.2) and X-pIRT to estimate how good each template is coupled with sequential elimination algorithm [2] (as in Shi et al. [37]) to select prompt-example pairs for evaluation in each round. In Figure 7 we compare our PE to the baseline TRIPLE-GSE [37] with a logistic regression performance predictor and the same three types of covariates (PE-OneHot corresponds to PE-Rasch in previous experiments). For all covariate choices, we show that using PromptEval for best-prompt identification results in lower regret, *i.e.*, the performance of the best template minus the performance of the chosen template. We include the full results for other benchmarks and also apply TRIPLE-GSE with an MLP in Appendix H.1.

H.1. Extra results for best-prompt identification

In Figures 8, 9, and 10, we can see the full results for MMLU, BBH, and LMentry. For all benchmarks, we can see that within each triple “PE”, “GTRIPLE-SE”, “TRIPLE-MLP-GSE”, the “PE” version always has some advantage with a lower regret.

The tuning and fitting process of the Multi-Layer Perceptron (MLP) classifier involves setting up a pipeline that includes

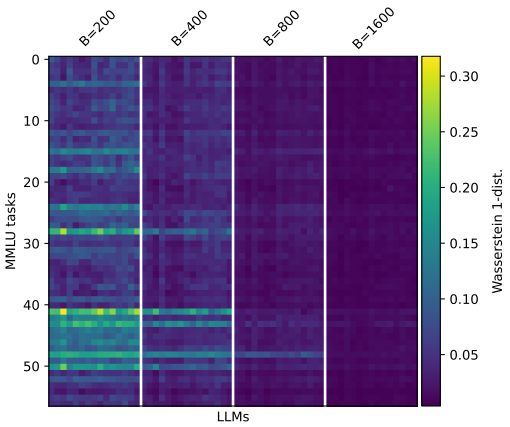

Figure 6: Estimation error for the MMLU tasks.

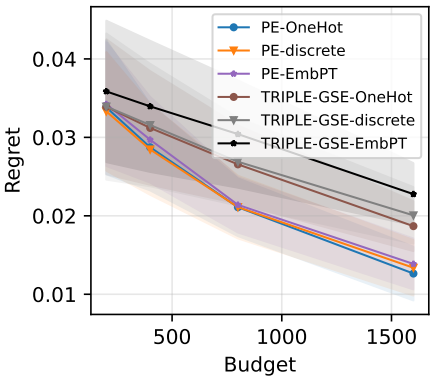

Figure 7: Best-prompt identification for MMLU

feature scaling and the MLP classifier itself, which has 30 neurons in its hidden layer. This process begins by defining a range of values for critical hyperparameters: the l2 regularization strength is tested over the range from 0.001 to 10, and the initial learning rate is tested over the range from 0.001 to 0.1. These values are systematically tested through cross-validation to determine the optimal combination. During this phase, cross-validation ensures that the model is evaluated on different subsets of the data to prevent overfitting and to ensure robust performance. Once the best hyperparameters are identified, the final model is trained on the entire dataset using these optimal settings, resulting in a well-tuned MLP classifier ready for deployment.

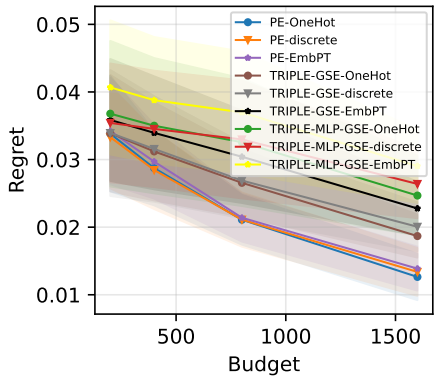

Figure 8: Best-prompt identification for MMLU

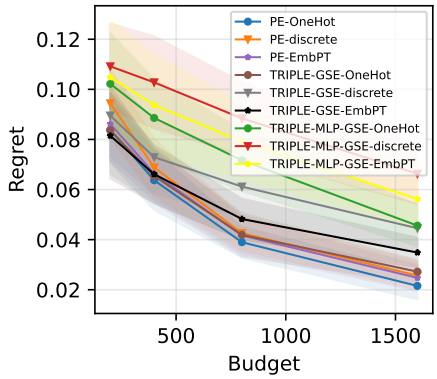

Figure 9: Best-prompt identification for BBH

I. Theoretical guarantees

In this section, we claim the consistency of the distribution and quantile estimators detailed in Algorithm 1 as $I, J \to \infty$. We prove a result for the case in which f_ψ and g_γ are linear/affine functions. Before we introduce our results we need to introduce some basic conditions. As an extra result, in Appendix I.1.1 we also show that our extended version of pIRT, X-pIRT, is uniformly consistent over all $i \in \mathcal{I}$, which can be useful beyond this work.

We start by assuming that the covariates are uniformly bounded.

Condition I.1. There is a universal constant $c > 0$ such that $\sup_{i\in\mathcal{I}} \|x_i\|_2, \sup_{j\in\mathcal{J}} \|z_j\|_2 < c$.

The next condition requires the number of unseen examples to increase sufficiently fast as $I, J \to \infty$, which is a realistic condition under the low-budget setup. Weaker versions of this condition are possible; we adopt this one because it makes our proof simpler.

Condition I.2. Assume (i) $m = |\mathcal{J} \setminus \mathcal{J}_i|$ is the same for all i's and grows to infinity and (ii) $\exp(\delta m)/I \to \infty$ as $I, J \to \infty$ for any $\delta > 0$.

The third condition requires the model we work with to be correctly specified and the maximum likelihood estimator of (ψ, γ) to be consistent as $I, J \to \infty$, *i.e.*, approach the true value. Evidently, $|\mathcal{E}|$ needs to grow to infinity as $I, J \to \infty$; nevertheless, it could be the case that $|\mathcal{E}|/(I \cdot J) \to 0$. When f_ψ and g_γ are linear/affine, the maximum likelihood procedure is equivalent to fitting a logistic regression model and, in that case, the convergence of $(\hat{\psi}, \hat{\gamma})$ is well-studied and holds under mild conditions when the dimensions of the covariates are fixed; see, for example, Fahrmeir and Kaufmann [12].

Condition I.3. The data point Y_{ij} is sampled from a Bernoulli distribution with mean $\sigma(\psi_0^\top x_i - \gamma_0^\top z_j)$ for some true global parameter values ψ_0 and γ_0. Moreover, we assume that $(\hat{\psi}, \hat{\gamma}) \to (\psi_0, \gamma_0)$ in probability as $I, J \to \infty$.

We now introduce the main result in Theorem I.4, which shows the consistency of the distribution and quantile functions estimators introduced in Algorithm 1. See Appendix I.1 for the proof.

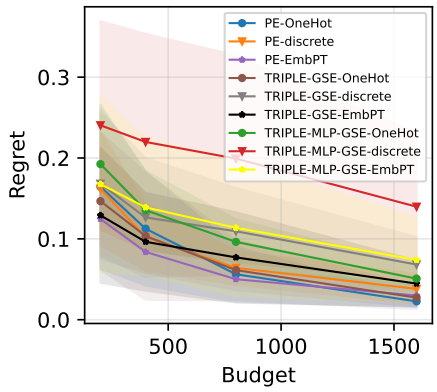

Figure 10: Best-prompt identification for LMentry

Theorem I.4. *Under conditions I.1, I.2, and I.3, it is true that*

$$\left|\hat{Q}_{\mathcal{I}}(p) - Q_{\mathcal{I}}(p)\right| \to 0 \text{ in probability as } I, J \to \infty \text{ for any } p \in [0,1],$$

and that

$$W_1(F, \hat{F}) \to 0 \text{ in probability as } I, J \to \infty,$$

where $W_1(F, \hat{F})$ is the Wasserstein 1-distance between the distributions F and $\hat{F}$.

I.1. Extra results and proofs

I.1.1. Consistency of X-pIRT

In Theorem I.5, we claim that the X-pIRT estimator is uniformly consistent over all $i \in \mathcal{I}$.

Theorem I.5. *Under conditions I.1, I.2, and I.3, it is true that*

$$\sup_{i \in \mathcal{I}} \left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{S}}] - S_i\right| \to 0 \text{ in probability as } I, J \to \infty.$$

A direct consequence of Theorem I.5 is that

$$\left|\tfrac{1}{I}\sum_{i \in \mathcal{I}} \hat{\mathbb{E}}[S_i \mid Y_{\mathcal{S}}] - \tfrac{1}{I}\sum_{i \in \mathcal{I}} S_i\right| \leq \tfrac{1}{I}\sum_{i \in \mathcal{I}} \left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{S}}] - S_i\right| \leq \sup_{i \in \mathcal{I}} \left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{S}}] - S_i\right| \to 0$$

in probability as $I, J \to \infty$. This means that the mean of predicted performances is also consistent if a practitioner wants to use it as a summary statistic.

The proof of Theorem I.5 is embedded in the proof of Theorem I.4.

I.1.2. Proof of Theorem I.4

For the following results, we denote $\psi^\top x_i$ as θ_i and $\gamma^\top z_j$ as β_j, and $\hat{\psi}^\top x_i$ as $\hat{\theta}_i$ and $\hat{\gamma}^\top z_j$ as $\hat{\beta}_j$. Moreover, if a sequence random variables (X_n) converge to 0 in distribution, we denote $X_n = o_P(1)$.

Lemma I.6. *Under Conditions I.1 and I.3, we have that $\sup_{i \in \mathcal{I}} |\hat{\theta}_i - \theta_i| = o_P(1)$ and $\sup_{j \in \mathcal{J}} |\hat{\beta}_j - \beta_j| = o_P(1)$ as $I, J \to \infty$.*

Proof. We prove that $\sup_{i \in \mathcal{I}} |\hat{\theta}_i - \theta_i| = o_P(1)$. The second statement is obtained in the same way.

See that

$$\sup_{i \in \mathcal{I}} |\hat{\theta}_i - \theta_i| = \sup_{x_i} |(\hat{\psi} - \psi)^\top x_i| \leq \sup_{x_i} \left\|\hat{\psi} - \psi\right\|_2 \|x_i\|_2 \leq c \left\|\hat{\psi} - \psi\right\|_2 = o_P(1)$$

as $I, J \to \infty$. Where the first inequality is obtained using the Cauchy–Schwarz inequality, the second is obtained using Condition I.1, and the last equality is a consequence of Condition I.3 and the continuous mapping theorem [34]. □

Lemma I.7. *Under Conditions I.1 and I.3, it is true that*

$$\sup_{i\in\mathcal{I}}\left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - \mathbb{E}[S_i \mid Y_{\mathcal{E}}]\right| = o_P(1) \text{ as } I, J \to \infty.$$

Proof. See that

$$\begin{aligned}
\sup_{i\in\mathcal{I}}\left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - \mathbb{E}[S_i \mid Y_{\mathcal{E}}]\right| &= \sup_{i\in\mathcal{I}} \frac{1-\lambda_i}{J-|\mathcal{J}_i|}\left|\sum_{j\notin\mathcal{J}_i}\sigma(\hat{\theta}_i-\hat{\beta}_j)-\sigma(\theta_i-\beta_j)\right| \\
&\leq \sup_{i\in\mathcal{I}} \frac{1-\lambda_i}{J-|\mathcal{J}_i|}\sum_{j\notin\mathcal{J}_i}\left|\sigma(\hat{\theta}_i-\hat{\beta}_j)-\sigma(\theta_i-\beta_j)\right| \\
&\leq \sup_{i\in\mathcal{I}} \frac{1-\lambda_i}{4(J-|\mathcal{J}_i|)}\sum_{j\notin\mathcal{J}_i}\left|\hat{\theta}_i-\hat{\beta}_j-\theta_i+\beta_j\right| \\
&\leq \frac{1-\inf_i\lambda_i}{4}\left(\sup_j|\hat{\theta}_i-\theta_i|+\sup_j|\hat{\beta}_j-\beta_j|\right) \\
&\leq \frac{1}{4}\left(\sup_i|\hat{\theta}_i-\theta_i|+\sup_j|\hat{\beta}_j-\beta_j|\right) \\
&= o_P(1)
\end{aligned}$$

where the third step is justified by the fact that σ is $1/4$-Lipschitz and the last step is justified by Lemma I.6. □

Lemma I.8. *Under Condition I.2, it is true that*

$$\sup_{i\in\mathcal{I}}|\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| = o_P(1) \text{ as } I, J \to \infty.$$

Proof. For an arbitrary $\epsilon > 0$, see that

$$\begin{aligned}
\mathbb{P}\left(\sup_{i\in\mathcal{I}}|\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| \geq \epsilon\right) &= \mathbb{P}\left(\bigcup_{i\in\mathcal{I}}\{|\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| \geq \epsilon\}\right) \\
&\leq \sum_{i\in\mathcal{I}}\mathbb{P}\left(|\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| \geq \epsilon\right) \\
&= \sum_{i\in\mathcal{I}}\mathbb{P}\left(\left|\frac{\lambda_i}{|\mathcal{J}_i|}\sum_{j\in\mathcal{J}_i}Y_{ij}+\frac{1-\lambda_i}{|\mathcal{J}\setminus\mathcal{J}_i|}\sum_{j\notin\mathcal{J}_i}\sigma(\theta_i-\beta_j)-\frac{1}{J}\sum_{j\in\mathcal{J}}Y_{ij}\right| \geq \epsilon\right) \\
&= \sum_{i\in\mathcal{I}}\mathbb{P}\left(\left|(1-\lambda_i)\frac{1}{|\mathcal{J}\setminus\mathcal{J}_i|}\sum_{j\notin\mathcal{J}_i}Z_{ij}\right| \geq \epsilon\right) \\
&\leq \sum_{i\in\mathcal{I}}\mathbb{P}\left(\left|\frac{1}{m}\sum_{j\notin\mathcal{J}_i}Z_{ij}\right| \geq \epsilon\right)
\end{aligned}$$

where $Z_{ij} \triangleq Y_{ij} - \sigma(\theta_i - \beta_j)$. Consequently, $|Z_{ij}| \leq 1$ and $\mathbb{E}[Z_{ij}] = 0$. Applying Hoeffding's inequality, we obtain

$$\begin{aligned}
\mathbb{P}\left(\sup_{i\in\mathcal{I}}|\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| \geq \epsilon\right) &\leq 2I\exp\left(-2m\epsilon^2\right) \\
&= 2\exp\left(\log I - 2\epsilon^2 m\right) \\
&= 2\exp\left(-\log(\exp(2\epsilon^2 m)/I)\right) \\
&\to 0
\end{aligned}$$

□

Lemma I.9. *Let $a_1, \cdots, a_n$ and $b_1, \cdots, b_n$ be two lists of real numbers and let a_i and b_j be the p-lower quantiles of those lists. Admit that there are m_1 a's lower than a_i, m_2 a's equal to a_i (besides a_i itself), and m_3 a's greater than a_i. Then, there are at least $m_1 + 1$ b's lower or equal to b_j and at least $m_3 + 1$ b's greater or equal to b_j.*

Proof. If a_i is the p-lower quantile of $\mathcal{A} = \{a_1, \cdots, a_n\}$, then by definition a_i is the lowest value in $\mathcal{A}$ such that

$$\underbrace{|\{a \in \mathcal{A} : a_i = a\}|}_{m_2+1} + \underbrace{|\{a \in \mathcal{A} : a_i > a\}|}_{m_1} \geq p \cdot n$$

Because a_i is the lowest value in $\mathcal{A}$ to achieve that, then $m_1 < p \cdot n$. This implies that there are at least $m_1 + 1$ values in $\mathcal{B} = \{b_1, \cdots, b_n\}$ lower or equal to b_j as it is the p-lower quantile of $\mathcal{B}$.

Finally, because $m_1 + m_2 + 1 \geq p \cdot n$, we know that $\mathcal{B}$ cannot have more than $m_1 + m_2$ values strictly lower than b_j, otherwise the p-lower quantile of $\mathcal{B}$ could not be b_j but some of those values. Therefore, $\mathcal{B}$ have at least $m_3 + 1$ values greater or equal to b_j. □

Lemma I.10. *Let $\hat{i}$ and i^* be indices in $\mathcal{I}$ such that $\hat{Q}(p) = \hat{\mathbb{E}}[S_{\hat{i}} \mid Y_S]$ and $Q(p) = S_{i^*}$ for an arbitrary fixed $p \in [0, 1]$. Under $\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - S_i| \leq \epsilon$ for an arbitrary $\epsilon > 0$, if $|\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}] - S_{i^*}| > 2\epsilon$, for some $i' \in \mathcal{I}$, then $\hat{i} \neq i'$. Consequently, if $\hat{i} = i'$ then $|\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}] - S_{i^*}]| \leq 2\epsilon$ under $\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - S_i| \leq \epsilon$.*

Proof. Define $\mathcal{A} \triangleq \{S_1, \cdots, S_i\}$ and assume that there are M_1 values in $\mathcal{A}$ lower than S_{i^*}, M_2 values equal to S_{i^*} (besides S_{i^*} itself), and M_3 values greater than S_{i^*}. If $|S_{i'} - S_{i^*}| > 2\epsilon$, for a certain index i', there are two possibilities: (i) $S_{i'} + 2\epsilon < S_{i^*}$ or (ii) $S_{i^*} + 2\epsilon < S_{i'}$. Under the event $\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - S_i| \leq \epsilon$, we have:

- If (i) holds, then there is at least $M_2 + M_3 + 1$ values of $\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$'s such that $\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}] < \hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$ (including $\hat{\mathbb{E}}[S_{i^*} \mid Y_{\mathcal{E}}]$). This implies that at most M_1 values of $\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$'s will be less or equal $\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}]$. By Lemma I.9, we know that $j' \neq \hat{j}$.
- If (ii) holds, then there is at least $M_1 + M_2 + 1$ values of $\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$'s such that $\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}] > \hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$ (including $\hat{\mathbb{E}}[S_{i^*} \mid Y_{\mathcal{E}}]$). This implies that at most M_3 values of $\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}]$'s will be greater or equal $\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}]$. By Lemma I.9, we know that $j' \neq \hat{j}$.

This means that under $\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - S_i| \leq \epsilon$ for an arbitrary $\epsilon > 0$, if $|\hat{\mathbb{E}}[S_{i'} \mid Y_{\mathcal{E}}] - S_{i^*}| > 2\epsilon$, for some $i' \in \mathcal{I}$, then $\hat{i} \neq i'$. □

Proof of Theorem I.4 (Part 1). Let $\hat{i}$ and i^* be indices in $\mathcal{I}$ such that $\hat{Q}(p) = \hat{\mathbb{E}}[S_{\hat{i}} \mid Y_S]$ and $Q(p) = S_{i^*}$. Notice that Lemma I.10 guarantees that $\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_S] - S_i| \leq \epsilon$ implies $|\hat{\mathbb{E}}[S_{\hat{i}} \mid Y_S] - S_{i^*}| \leq 2\epsilon$, for an arbitrary $\epsilon > 0$. Consequently,

$$\mathbb{P}\left(|\hat{\mathbb{E}}[S_{\hat{i}} \mid Y_S] - S_{i^*}| \leq 2\epsilon\right) \geq \mathbb{P}\left(\sup_{i \in \mathcal{I}} |\hat{\mathbb{E}}[S_i \mid Y_S] - S_i| \leq \epsilon\right) = 1 + o(1)$$

because

$$\sup_{i \in \mathcal{I}} \left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - S_i\right| \leq \sup_{i \in \mathcal{I}} \left|\hat{\mathbb{E}}[S_i \mid Y_{\mathcal{E}}] - \mathbb{E}[S_i \mid Y_{\mathcal{E}}]\right| + \sup_{i \in \mathcal{I}} |\mathbb{E}[S_i \mid Y_{\mathcal{E}}] - S_i| = o_P(1) \text{ as } I, J \to \infty$$

holds by lemmas I.7 and I.8. Because $\epsilon > 0$ is arbitrary, we have that $|\hat{Q}(p) - Q(p)| = o_P(1)$.

□

Proof of Theorem I.4 (Part 2). We start this proof by showing that

$$|\hat{Q}(U) - Q(U)| = o_P(1)$$

with $U \sim \text{Unif}[0, 1]$ independent of $\hat{Q}$ and Q.

For an arbitrary $\epsilon > 0$, see that

$$\begin{aligned}\lim_{I,J\to\infty} \mathbb{P}(|\hat{Q}(U) - Q(U)| > \epsilon) &= \lim_{I,J\to\infty} \mathbb{E}\left[\mathbb{P}(|\hat{Q}(U) - Q(U)| > \epsilon \mid U)\right] \\ &= \mathbb{E}\left[\lim_{I,J\to\infty} \mathbb{P}(|\hat{Q}(U) - Q(U)| > \epsilon \mid U)\right] \\ &= 0\end{aligned}$$

where the second equality is justified by the Dominated Convergence Theorem [34] and the last one is justified by $|\hat{Q}(p) - Q(p)| = o_P(1)$. Now, we see that

$$\begin{aligned}\lim_{I,J\to\infty} \mathbb{E}[W_1(F, \hat{F})] &= \lim_{I,J\to\infty} \mathbb{E}\left[\int_0^1 |Q(t) - \hat{Q}(t)| \mathrm{d}t\right] \\ &= \lim_{I,J\to\infty} \mathbb{E}\left[|\hat{Q}(U) - Q(U)|\right] \\ &= 0\end{aligned}$$

where the last step is justified by Fubini's Theorem [34], $|\hat{Q}(U) - Q(U)| = o_P(1)$, and the Lebesgue Dominated Convergence Theorem [34]. For an arbitrary $\epsilon > 0$ and applying Markov's inequality, we get

$$\lim_{I,J\to\infty} \mathbb{P}(W_1(F, \hat{F}) > \epsilon) \leq \frac{1}{\epsilon} \lim_{I,J\to\infty} \mathbb{E}[W_1(F, \hat{F})] = 0$$

□

J. Details MMLU data

Algorithm 3 for automatically generating templates can be seen as a graph traversal of a template graph, whose nodes are defined by which features they have: a separator SEP, a space SPA, and an operator OP. By traversing this graph, we can collect unique templates that can used in the evaluation of LLMs on tasks.

Algorithm 3: TemplateGeneration

1 **Input:** Base prompt template features: Separator SEP, Space SPA, Operator OP.

2 **Output:** Prompt templates.

3 From template agenda, pop a template. Swap SEP with another SEP, add to templates. Swap SPA with another SPA, add to templates. Swap OP with another OP, add to templates. Add the generated templates to the agenda.

4 **return** generated templates.

Next, we utilize the unitxt [3] preprocessing library to build custom datasets with the generated templates. Standardized and accurate evaluation is then carried out via the LM-Eval-Harness [13] evaluation library.

K. Analysis of prompt sensitivity on MMLU

Prior work reports strong sensitivity of LLMs to spurious prompt template changes (see Section A). For example, Sclar et al. [36] observe performance changes of up to 80% for Natural Instructions tasks [45] due to template changes. Despite its popularity, no such analysis exists for the MMLU dataset to date. We here provide an in-depth analysis of MMLU prompt sensitivity.

Performance spread When averaged across subjects, we observe relatively small performance spreads per LLM

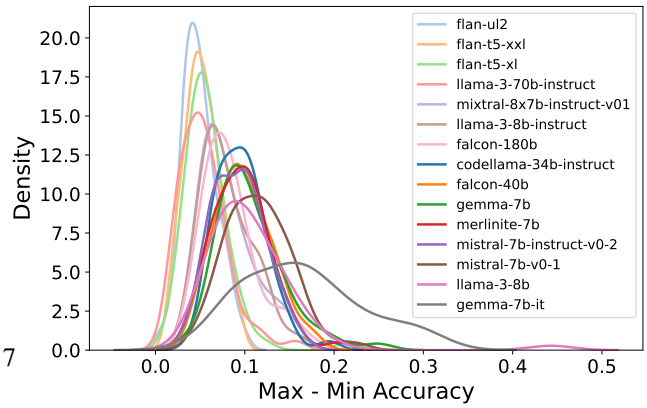

Figure 11: Accuracy spread across 57 subjects.

compared to other datasets in the literature (see Figure 12 in the Appendix K.1). For example, we can consistently identify `Llama-3-70B-Instruct` as the best performing model, independent of the prompt template. On the other hand, scores within individual subjects are highly inconsistent. Figure 11 shows the distribution of prompt spreads (max-min acc.) across subjects per LLM. Most LLMs demonstrate a significant average spread of around 10% at the subject level.

Template consistency In practice, having consistently performing templates is highly relevant *within a single LLM* or *across LLMs* for the same subject. To evaluate the template consistency, we rank template performances either across subjects or across LLMs to then calculate the agreement across those rankings using Kendall's W [19, inspired by Mizrahi et al. 28].

Within LLMs, we observe that `Gemma-7B-it` has a notably higher Kendall's W of 0.45 than any other model, meaning a fixed set of prompts performs best across many subjects (for full results, see Table 1 in the Appendix). Across LLMs, we do not observe high correlations within any of the subjects (see Figure 13 in Appendix K.1). Hence, similar to previous findings [*e.g.* 36], we do not identify any coherent template preferences across LLMs (for detailed results, see Appendix K.1).

K.1. Details MMLU spread analysis

Figure 12 depicts the performance of LLMs on the whole MMLU.

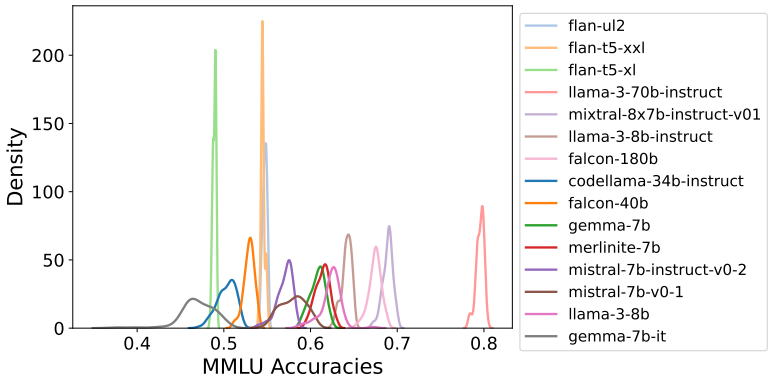

Figure 12: MMLU accuracy (all 57 subjects).

To correlate the ranks from different judges, we can use Kendall's W. Kendall's W [19] ranges from 0 (no agreement) to 1 (perfect agreement) and is calculated as $W = \frac{12S}{m^2(n^3-n)}$, where S is the sum of squared deviations of the total ranks from the mean rank, m is the number of rankers, and n is the number of objects ranked. In our case, we first have MMLU subjects ranking prompt templates, and then we have LLMs ranking prompt templates.

In Figure 13, we see the distribution of Kendall's W for subjects ranking templates. The correlation is not significant, with the highest W around 0.25. This suggests that there is no "best" prompt for a subject.

In Table 1, we see the values of Kendall's W for each model. For most models, the W value is not high, but for gemma-7b and mistral-7b-v0-1, the value of W is 0.45 and 0.35, respectively. Curiously, both of the top-ranked prompt templates have lots of commas. The best-ranked prompt is **"The, following, are, multiple, choice, questions, (with, answers), about, topic], question], Answers], choices], Answer]"**. Interestingly, the comma separation of each word or phrase in this prompt template may aid the model in parsing and effectively understanding the different components of the prompt structure.

Figure 14 illustrates sensitivity for llama-3-8b, gemma-7b, and merlinite-7b, respectively. On the template graph, a distance 1 means templates differ by only 1 feature, a distance 2 means templates differ by 2 features, etc. We see that there is no significant correlation between template distance and the accuracy spread. In the cases of gemma-7b and merlinite-7b, the accuracy spread for templates with smaller distance seems to be smaller, possibly implying that the template graph for these models is smooth.

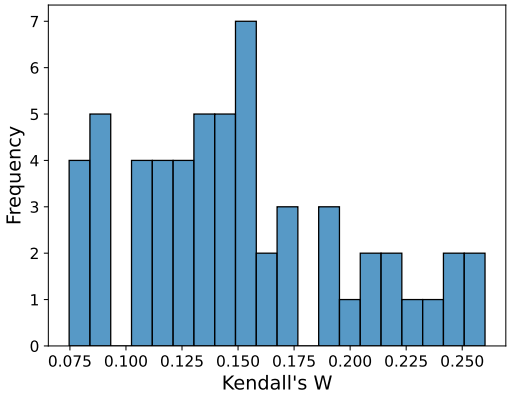

Figure 13: Kendall's W per MMLU subject: Here we see a distribution of Kendall's W over the 57 subjects of MMLU.

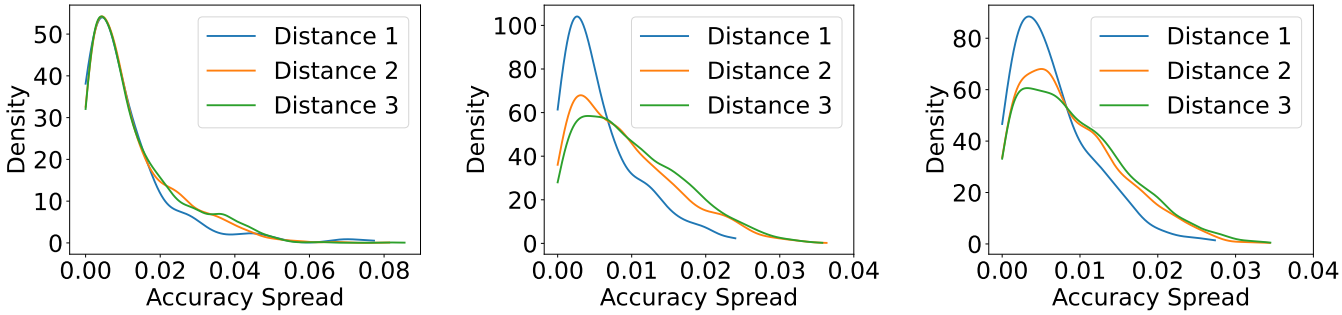

Figure 14: Model sensitivity for llama-3-8b, gemma-7b, and merlinite-7b.

L. BERT fine-tuning details

L.1. Model

We augment the BERT model by extending its input embeddings by $|J|$ [Example ID] tokens which we use to feed information about the example identity to the model. Additionally, we add a linear downward projection (*d = 25*) on top of the final BERT layer to reduce the dimensionality of the resulting covariates.

L.2. Training data

To obtain training examples, we concatenate all prompting templates with all [Example ID] tokens giving us $|I| \times |J|$ model inputs (giving us the following dataset sizes for the respective benchmarks: BBH 209,280; LMentry 175,776; and MMLU 1,121,568). Labels consist of vectors of correctness scores y_{ij} from the LLMs in the training set, making the training task a multi-label binary classification problem. We train on an iid split of half of the LLMs at a time and test on the other half. Additionally, the training data are split along the example axis into an 80% training and 20% validation set.

L.3. Hyperparameters

We run a small grid search over different plausible hyperparameter settings and settle on the following setup: We employ the Adam optimizer [20] with an initial learning rate of 2e-5 and a weight decay of 1e-5. The learning rate undergoes a linear warm-up over 200 steps, followed by exponential decay using the formula $lr_{currnt} = \gamma^s \cdot lr_{init}$, where s is the number of steps after the warmup phase and the decay factor γ is set to 0.99995. We train with a batch size of 96.

Table 1: Kendall's W per LLM

Model	Kendall's W
meta-llama/llama-3-8b-instruct	0.126027
meta-llama/llama-3-8b	0.252835
meta-llama/llama-3-70b-instruct	0.101895
mistralai/mistral-7b-instruct-v0-2	0.219841
mistralai/mistral-7b-v0-1	0.345592
mistralai/mixtral-8x7b-instruct-v01	0.131487
codellama/codellama-34b-instruct	0.287066
ibm-mistralai/merlinite-7b	0.146411
google/gemma-7b-it	**0.445478**
google/gemma-7b	0.179373
google/flan-t5-xl	0.066501
google/flan-t5-xxl	0.056257
google/flan-ul2	0.109076
tiiuae/falcon-180b	0.165600
tiiuae/falcon-40b	0.100173

M. Heuristics for discrete features

For the BBH and LMentry benchmarks, we use the following heuristics to construct feature representations of prompt templates.

Table 2: Overview of Discrete Features

Category	Feature Name	Description
Casing Features	All Caps Words	Count of all uppercase words
	Lowercase Words	Count of all lowercase words
	Capitalized Words	Count of words with the first letter capitalized
Formatting Features	Line Breaks	Count of line breaks
	Framing Words	Count of capitalized or numeric words before a colon
Special Characters Features	Colon (:)	Count of ':'
	Dash (-)	Count of '-'
	Double Bar (——)	Count of '——'
	Separator token	Count of '¡sep¿'
	Double Colon (::)	Count of '::'
	Parenthesis Left (()	Count of '('
	Parenthesis Right ())	Count of ')'
	Quotation (")	Count of '"'
	Question Mark (?)	Count of '?'
Length Feature	Space Count	Count of spaces