TINYBENCHMARKS: EVALUATING LLMS WITH FEWER EXAMPLES

Felipe Maia Polo[♠], **Lucas Weber**[♦], **Leshem Choshen**[▲★]
Yuekai Sun[♠], **Gongjun Xu**[♠], **Mikhail Yurochkin**[★♣]
[♠] University of Michigan [♦] University of Pompeu Fabra [▲] MIT
[★] IBM Research [♣] MIT-IBM Watson AI Lab

ABSTRACT

The versatility of large language models (LLMs) led to the creation of diverse benchmarks that thoroughly test a variety of language models' abilities. These benchmarks consist of tens of thousands of examples making evaluation of LLMs very expensive. In this paper, we investigate strategies to reduce the number of evaluations needed to assess the performance of an LLM on several key benchmarks. For example, we show that to accurately estimate the performance of an LLM on MMLU, a popular multiple-choice QA benchmark consisting of 14K examples, it is sufficient to evaluate this LLM on 100 curated examples. We release evaluation tools and tiny versions of popular benchmarks: Open LLM Leaderboard, MMLU, HELM, and AlpacaEval 2.0. Our empirical analysis demonstrates that these tools and tiny benchmarks are sufficient to reliably and efficiently reproduce the original evaluation results. Please check the complete and updated version of this work in https://arxiv.org/abs/2402.14992.

1 INTRODUCTION

Large Language Models (LLMs) have demonstrated remarkable abilities to solve a diverse range of tasks [7]. Quantifying these abilities and comparing different LLMs became a challenge that led to the development of several key benchmarks, e.g., MMLU [15], Open LLM Leaderboard [4], HELM [23], and AlpacaEval [22]. These benchmarks are comprised of hundreds or thousands of examples, making the evaluation of modern LLMs with billions of parameters computationally, environmentally, and financially very costly. Our work presents different ways of making benchmark evaluation more efficient by selecting small sets of curated examples. In Figure 1 we demonstrate the efficacy of an evaluation strategy on MMLU, where we compare accuracy estimates obtained from evaluating LLMs on a curated subset of 100 examples ($< 1\%$ of the examples) to accuracy on all of MMLU, achieving average estimation error under 2%.

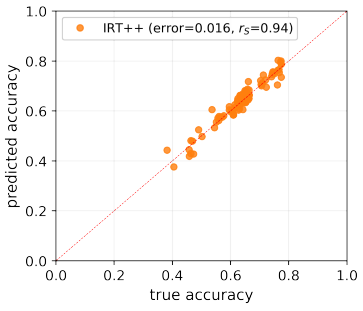

Figure 1: Estimating accuracy on MMLU (true accuracy) using 100 curated examples (predicted accuracy). IRT++, our best-performing evaluation strategy, predicts the accuracy of recent LLMs released between December 30th and January 18th within 1.6% of their true accuracy on all of MMLU (14K examples).

We consider a range of evaluation strategies (§3) such as stratified random sampling, selecting a small but representative set of examples using clustering algorithms, and applying item response theory (IRT) methods for performance estimation. We present an extensive evaluation of these strategies on four popular benchmarks (§5): Open LLM Leaderboard [4], MMLU [15], HELM [23], and AlpacaEval 2.0 [22]. Our goal is to assess the effectiveness of estimating the performance of LLMs on these benchmarks using a limited number of examples for evaluation. Overall, we conclude that 100 curated examples per scenario are enough to reliably estimate the performance of various LLMs, within about 2% error on average.

Based on our findings we release tiny (100 examples per scenario) versions of every considered benchmark and IRT-based tools for further improving the performance estimation. In Appendix A, we talk in detail about related work.

2 PROBLEM STATEMENT

In this section, we describe in detail the setup we work on and what are our objectives. Consider that a benchmark is composed of scenarios and possibly sub-scenarios. For example, MMLU and HellaSwag are examples of scenarios[1] of both the Open LLM Leaderboard and HELM, while MMLU has different sub-scenarios like "marketing", "elementary mathematical", and so on. Furthermore, each scenario (or sub-scenario) is composed of examples (analogous to "items" in the IRT literature) that are small tests to be solved by the LLMs–these examples range from multiple-choice questions to text summarization tasks. Our final objective is to estimate the performance of LLMs in the full benchmark, which is given by the average of the performances in individual scenarios (Open LLM Leaderboard, MMLU, AlpacaEval 2.0) or mean-win-rate (HELM). We achieve this objective by first estimating the performance of LLMs in individual scenarios and then aggregating scores. When scenarios have sub-scenarios, it is usually the case that the scenario performance is given by a simple average of sub-scenarios performances. The main concern is that each scenario/sub-scenario is composed of hundreds or thousands of examples, making model evaluation costly.

In this work, for a fixed benchmark, we denote the set of examples of each scenario j as $\mathcal{I}_j$, implying that the totality of examples in the benchmark is given by $\mathcal{I} = \cup_j \mathcal{I}_j$. When an LLM l interacts with an example $i \in \mathcal{I}_j$, the system behind the benchmarks generates a score that we call "correctness" and denote as Y_{il}. In all the benchmarks we consider in this work, the correctness is either binary, *i.e.*, $Y_{il} \in \{0, 1\}$ (incorrect/correct), or bounded, *i.e.*, $Y_{il} \in [0, 1]$, denoting a degree of correctness. The second case is applied in situations in which, for instance, there might not be just one correct answer for example i. To simplify the exposition in the text, we assume that the score for LLM l in scenario j is just the simple average of the correctness of all items in that scenario, that is, $\frac{1}{|\mathcal{I}_j|}\sum_{i\in\mathcal{I}_j} Y_{il}$. That is not true when different sub-scenarios have different numbers of examples; in that case, one would just have to use a weighted average instead, to make sure every sub-scenario is equally important (in the experiments, we consider this case). Our objective when evaluating a model l is to choose a small fraction of examples $\widehat{\mathcal{I}}_j \subset \mathcal{I}_j$, compute the correctness of model l in every example of $\widehat{\mathcal{I}}_j$, and then use the available data to estimate what would be the average score of that model in $\mathcal{I}_j$, *i.e.*, $\frac{1}{|\mathcal{I}_j|}\sum_{i\in\mathcal{I}_j} Y_{il}$. In the next section, we describe strategies on how $\widehat{\mathcal{I}}_j$ can be chosen and how the overall score can be estimated.

3 SELECTING EVALUATION EXAMPLES

In this section, we describe strategies on how to select examples from a fixed scenario j, *i.e.*, $\mathcal{I}_j$, obtaining $\widehat{\mathcal{I}}_j \subset \mathcal{I}_j$ described in Section 2. Ideally, the set of selected examples should be representative of the whole set of items in scenario j, that is,

$$\sum_{i\in\widehat{\mathcal{I}}_j} w_i Y_{il} \approx \frac{1}{|\mathcal{I}_j|}\sum_{i\in\mathcal{I}_j} Y_{il}, \tag{3.1}$$

for nonnegative weights $\{w_i\}_{i\in\widehat{\mathcal{I}}_j}$ such that $\sum_{i\in\widehat{\mathcal{I}}_j} w_i = 1$. In the next paragraphs, we describe two possible ways of obtaining $\widehat{\mathcal{I}}_j$ and $\{w_i\}_{i\in\widehat{\mathcal{I}}_j}$.

Stratified random sampling. In some settings [e.g., classifiers 16], it is useful to perform stratified random sampling – subsample examples, but ensure representation of certain groups of data. Using sub-scenarios as the strata for stratified random sampling was proposed by Perlitz et al. [32] when sub-sampling examples from HELM scenarios. The authors showed that this is an effective way of sampling examples without too much loss on the ability to rank LLMs by performance. Examples should be randomly selected from sub-scenarios (with uniform probability) in a way such that the difference in number of examples sampled for two distinct subscenarios is minimal (≤ 1). The rationale behind this method is that, for an effective evaluation, sub-scenarios should be equally represented. Overall, the weights are $w_i = 1/|\widehat{\mathcal{I}}_j|$ for all $i \in \widehat{\mathcal{I}}_j$.

Clustering. Assessing the performance of LLMs on a randomly sampled subset of examples suffers from extra uncertainty in the sampling process, especially when the number of sampled examples is small. Instead, we consider selecting a subset of representative examples using clustering. Vivek

[1]We consider MMLU and AlpacaEval as a single scenario each.

et al. [42] proposed to cluster examples based on the confidence of models in the correct class corresponding to these examples. Representative examples, from these clusters, which they call "anchor points", can then be used to evaluate models on classification tasks more efficiently. We adapt their clustering approach to a more general setting, allowing us to extract such anchor points for MMLU, AlpacaEval 2.0, and all scenarios of the Open LLM Leaderboard and HELM.

To refine the selection of anchor points, two main strategies are proposed. The first strategy employs K-Means clustering based on the correctness scores of models in the training set across examples, aiming to identify a small number of examples (anchor points) that are predictive of model performance on a larger set. This approach, while straightforward and effective, is noted for potential vulnerabilities to distribution shifts and the curse of dimensionality in large training sets. The second strategy introduces a more nuanced approach using item response theory (IRT) parameter estimates, detailed in Section 4, to represent examples. This method addresses the limitations of the first by reducing dimensionality and potentially improving robustness against distribution shifts when the IRT model is reasonable in describing reality.

4 Better performance estimation with IRT

In this section, we propose ways of enhancing performance estimates by using IRT models. We start by discussing the case where $Y_{il} \in \{0, 1\}$, that is, the l responds to the example $i \in \mathcal{I}$ correctly or not. We discuss the case where $Y_{il} \in [0, 1]$ in Appendix B.

The IRT model. The two-parameter multidimensional IRT model assumes that the probability of the LLM j getting example i correctly is given by

$$p_{il} \triangleq \mathbb{P}(Y_{il} = 1 \mid \theta_l, \alpha_i, \beta_i) = \frac{1}{1+\exp(-\alpha_i^\top \theta_l + \beta_i)}, \tag{4.1}$$

where $\theta_l \in \mathbb{R}^d$ denotes the unobserved abilities of LLM l, while $\alpha_i \in \mathbb{R}^d$ dictates which dimensions of θ_l are required from model l to respond to example i correctly. In this formulation, $\beta_i \in \mathbb{R}$ can be viewed as a bias term that regulates the probability of correctness when $\theta_l = 0$. We use IRT parameter estimates as example representations referred to in Section 3. Specifically, we take $E_i = (\widehat{\alpha}_i, \widehat{\beta}_i)$, where $\widehat{\alpha}_i$ and $\widehat{\beta}_i$ are point estimates for the parameters of example i. We introduce two estimators for the performance of an LLM and describe model fitting in Appendix B.

IRT-based LLM performance estimation. Assume that we are interested in estimating the performance of a model $l \notin \mathcal{L}_{tr}$ on scenario j and that point estimates of example parameters, $(\widehat{\alpha}_i, \widehat{\beta}_i)$, have been computed for all examples in all scenarios, including examples $i \in \mathcal{I}_j$. Formally, we are interested in approximating

$$Z_{jl} \triangleq \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} Y_{il} \tag{4.2}$$

Now, assume that we have run model l on a subset of examples from scenario j, obtaining responses $\{Y_{i_0 l}, \cdots, Y_{i_k l}\}$ for the examples $\widehat{\mathcal{I}}_j = \{i_0, \cdots, i_k\}$. Let $\widehat{\theta}_l$ denote the estimate for θ_l after observing $\widehat{\mathcal{I}}_j$ and possibly a bigger set of examples coming from different scenarios. To obtain the estimate, we maximize the log-likelihood of the freshly observed data with respect to θ_l, fixing examples' parameters. This procedure is equivalent to fitting a logistic regression model.

Because Z_{jl} is a random variable, we approximate it by estimating the conditional expectation

$$\mathbb{E}[Z_{jl} \mid Y_{i_0 l}, \cdots, Y_{i_k l}] = \frac{\hat{\lambda}}{|\widehat{\mathcal{I}}_j|} \sum_{i \in \widehat{\mathcal{I}}_j} Y_{il} + \frac{1-\hat{\lambda}}{|\mathcal{I}_j \setminus \widehat{\mathcal{I}}_j|} \sum_{i \in \mathcal{I}_j \setminus \widehat{\mathcal{I}}_j} p_{il}$$

where $\hat{\lambda} = |\widehat{\mathcal{I}}_j| / |\mathcal{I}_j| \in [0, 1]$ is a weight that gives more or less importance to the observed set $\widehat{\mathcal{I}}_j$ in the performance computation depending on how big that set is. The probability $p_{il} = \mathbb{P}(Y_{il} = 1 \mid \theta_l, \alpha_i, \beta_i)$ is given by the IRT model in Equation 4.1. The estimator for the conditional expectation is then given by

$$\hat{Z}_{jl}^{\text{p-IRT}} \triangleq \frac{\hat{\lambda}}{|\widehat{\mathcal{I}}_j|} \sum_{i \in \widehat{\mathcal{I}}_j} Y_{il} + \frac{1-\hat{\lambda}}{|\mathcal{I}_j \setminus \widehat{\mathcal{I}}_j|} \sum_{i \in \mathcal{I}_j \setminus \widehat{\mathcal{I}}_j} \hat{p}_{il} \tag{4.3}$$

where $\hat{p}_{il} \triangleq \mathbb{P}(Y_{il} = 1 \mid \widehat{\theta}_l, \widehat{\alpha}_i, \widehat{\beta}_i)$. We call the estimator in 4.3 by Performance-IRT (p-IRT) estimator. The idea behind p-IRT is that we can estimate the performance of a model on unseen data making use of the IRT model. This is especially useful if we can fit $\widehat{\theta}_l$ using data from many scenarios: even though we observe just a few samples per scenario, p-IRT will leverage the whole available data, permitting better estimates for the performance of the LLM for all scenarios. The

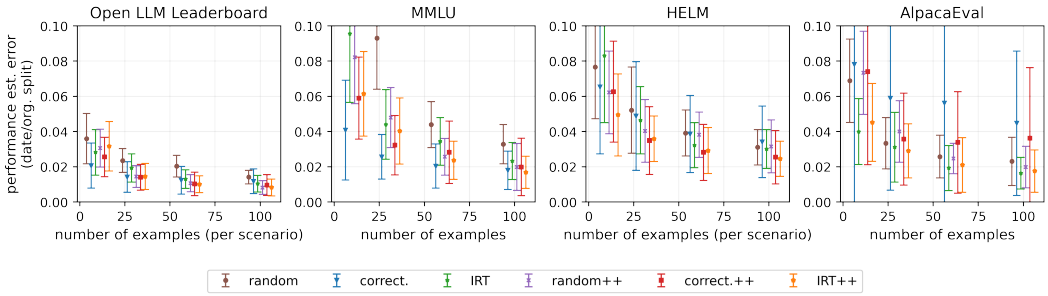

Figure 2: Performance estimation error per benchmark (columns) tested on recent LLMs for increasing number of evaluation examples. 100 examples per scenario are sufficient to achieve ≈2% average performance estimation error across benchmarks and evaluated LLMs. This corresponds to 600 out of 29K examples for Open LLM Leaderboard, 100 out of 14K examples for MMLU, 1500 out of 20K examples for HELM, and 100 out of 800 examples for AlpacaEval 2.0.

estimator p-IRT has low variance when $\widehat{\theta}_l$ is obtained from a large dataset and a small bias if the IRT model is reasonably specified.

We note two limitations of p-IRT that can hinder its effectiveness in practice. First, it does not promptly allow sample weighting, limiting its use of anchor points; second, if the predicted probabilities $\hat{p}_{il}$'s are inaccurate, *e.g.*, because of model misspecification, then the performance of p-IRT will deteriorate. In Appendix B, we introduce our final performance estimator, the Generalized p-IRT (also referenced as gp-IRT or IRT++ in the experiments) which is given by a convex combination of p-IRT and the estimator in equation 3.1.

5 Assessing Evaluation Strategies

We assess the ability of the considered evaluation strategies to estimate the performance of LLMs on four popular benchmarks: HuggingFace's Open LLM Leaderboard [4], MMLU [15], For HELM [23], AlpacaEval 2.0 [22]. In Appendix C, we describe all benchmarks and their data in detail. For a given LLM and a benchmark, each evaluation strategy estimates the performance using the evaluation results of this LLM on a given number of examples. We then compare this estimate to the true value, i.e., the performance of this LLM on the complete benchmark. We use publicly available evaluation data for which we split the models into "train" and "test". Evaluation of the train models is used to find the anchor points and fit the IRT model. The ability to predict performance is measured on the test set of models. We consider two train-test split scenarios: (i) random split and (ii) by date, i.e. using the most recent models for testing. The latter split better represents practical use cases.

Evaluation strategies. We consider 3 strategies presented in §3 for selecting a subset of examples for efficient evaluation: "random" for stratified random sampling, "correctness" for clustering correctness of models in the train set, and "IRT" for clustering the example representations obtained from the IRT model fit on the train set. For each strategy, we evaluate the vanilla variation, i.e., simply using the performance of a test LLM on the (weighted) set of selected examples to estimate its performance on the full benchmark, and "++" variation that adjusts this estimate using the IRT model as described in equation B.1. In total, we assess six evaluation strategies. Results are averaged over 5 restarts.

Key findings. We investigate the effectiveness of strategies as we increase the number of examples available for evaluating test LLMs. Results for non-random splits are presented in Figure 2 (full results in Figure 4 and Figure 9 for Spearman's rank correlations). Our approach to reducing evaluation costs is *effective*. The best-performing strategies achieve estimation error within 2% on all benchmarks with 100 examples or less per dataset or scenario. For example, for MMLU this reduces the evaluation cost by a factor of 140 (from 14k to 100). For Open LLM Leaderboard even 30 examples per scenario is enough, reducing the evaluation cost by a factor of 160 (from 29K to 180). The IRT-based methods ("IRT" and "IRT++") perform consistently well across benchmarks and train-test splits. Thus we use the IRT-based anchor examples to construct tiny versions (100 examples per scenario) of each of the benchmarks and release them along with the gp-IRT tool (code and pre-trained IRT model) for efficient evaluation of future LLMs. See demonstration[2].

[2] https://colab.research.google.com/drive/1txjUVWVFNEVue70thf8mlWVmB400gApk?usp=sharing

In Appendix E we conduct an exploratory analysis of the examples comprising tinyMMLU.

REFERENCES

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Neel Alex, Eli Lifland, Lewis Tunstall, Abhishek Thakur, Pegah Maham, C Jess Riedel, Emmie Hine, Carolyn Ashurst, Paul Sedille, Alexis Carlier, et al. Raft: A real-world few-shot text classification benchmark. *arXiv preprint arXiv:2109.14076*, 2021.

[3] Xinming An and Yiu-Fai Yung. Item response theory: What it is and how you can use the irt procedure to apply it. *SAS Institute Inc*, 10(4):364–2014, 2014.

[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] Stella Biderman, Hailey Schoelkopf, Quentin G. Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. *ArXiv*, abs/2304.01373, 2023. URL https://api.semanticscholar.org/CorpusID:257921893.

[6] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019. URL http://arxiv.org/abs/1903.04561.

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

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

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

[10] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*, 2018.

[11] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[12] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[13] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[14] Víctor Elvira, Luca Martino, and Christian P Robert. Rethinking the effective sample size. *International Statistical Review*, 90(3):525–550, 2022.

[15] 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.

[16] Namit Katariya, Arun Iyer, and Sunita Sarawagi. Active evaluation of classifiers on large datasets. In *2012 IEEE 12th International Conference on Data Mining*, pages 329–338, 2012. doi: 10.1109/ICDM.2012.161.

[17] Neal M Kingston and Neil J Dorans. The feasibility of using item response theory as a psychometric model for the gre aptitude test. *ETS Research Report Series*, 1982(1):i–148, 1982.

[18] Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[19] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[20] 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.

[21] John Patrick Lalor and Pedro Rodriguez. py-irt: A scalable item response theory library for python. *INFORMS Journal on Computing*, 35(1):5–13, 2023.

[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] 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.

[24] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

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

[26] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1015.

[27] Felipe Maia Polo and Renato Vicente. Effective sample size, dimensionality, and generalization in covariate shift adaptation. *Neural Computing and Applications*, 35(25):18187–18199, 2023.

[28] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[29] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[30] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

[31] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640: 660, 2016.

[32] 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.

[33] Nancy S Petersen et al. Using item response theory to equate scholastic aptitude test scores. 1982.

[34] 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.

[35] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[36] Wheyming Tina Song. Minimal-mse linear combinations of variance estimators of the sample mean. In *1988 Winter Simulation Conference Proceedings*, pages 414–421. IEEE, 1988.

[37] 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.

[38] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[41] 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.

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

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

[44] 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.

[45] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

A Related work

Efficient benchmarking of LLMs Multi-dataset benchmarks were introduced to the field of NLP with the advent of pre-trained models [e.g. 43], and constantly evolved in lockstep with language model capabilities [37]. The ever-increasing size of models and datasets consequently led to high evaluation costs, triggering changes in reported evaluation to accommodate the costs [5]. Ye et al. [44] considered reducing the number of *tasks* in Big-bench [37]. Perlitz et al. [32] found that evaluation on HELM [23] relies on diversity across datasets, but the number of examples currently used is excessive. We adopt their stratified sampling approach as one of the efficient evaluation strategies. Vivek et al. [42] proposed clustering evaluation examples based on models' confidence in the correct class for faster evaluation on classification tasks. One of the approaches we consider is based on an adaptation of their method to popular LLM benchmarks with more diverse tasks.

Item response theory (IRT) IRT [9, 40, 8, 25] is a well-established set of statistical models used in psychometrics to measure the latent abilities of individuals through standardized testing [3, 17, 33] (GRE, SAT, *etc.*). Even though IRT methods have been traditionally used in psychometrics, they are becoming increasingly popular among researchers in the fields of artificial intelligence and natural language processing (NLP). For instance, Lalor et al. [20] propose using IRT's latent variables to measure language model abilities, Vania et al. [41] employs IRT models in the context of language models benchmarking to study saturation (un-discriminability) of commonly used benchmarks, and Rodriguez et al. [34] study several applications of IRT in the context of language models, suggesting that IRT models can be reliably used to: predict responses of LLMs in unseen items, categorize items (*e.g.*, according to their difficulty/discriminability), and rank models. However, to the best of our knowledge, IRT has not been used for performance estimation in the context of efficient benchmarking of LLMs. We explore this new path.

B More details about IRT and estimators

The generalized p-IRT (gp-IRT) estimator. Our final estimator builds upon p-IRT to overcome its limitations. Assume that the estimators in equations 3.1 and 4.3 are obtained as a first step after the collection of examples in $\widehat{\mathcal{I}}_j$. The idea is to compute a third estimator $\hat{Z}_{jl}^{\text{gp-IRT}}$ given by a convex combination of the first two

$$\hat{Z}_{jl}^{\text{gp-IRT}} \triangleq \lambda \sum_{i \in \widehat{\mathcal{I}}_j} w_i Y_{il} + (1-\lambda)\hat{Z}_{jl}^{\text{p-IRT}} \tag{B.1}$$

where λ is a number in $[0, 1]$ that is chosen to optimize the performance of that estimator. To choose λ, we first note that using random sampling (or anchor points) implies low bias but potentially high variance (when $\widehat{\mathcal{I}}_j$ is small) for $\sum_{i \in \widehat{\mathcal{I}}_j} w_i Y_{il}$. As $\widehat{\mathcal{I}}_j$ grows, its variance decreases. On the other hand, the variance of $\hat{Z}_{jl}^{\text{p-IRT}}$ is small, especially when $\widehat{\theta}_l$ is fitted with data from many scenarios, but its bias can be high when the IRT model is misspecified and does not vanish with the growing sample size. Thus, good choice of λ increases with $\widehat{\mathcal{I}}_j$.

We choose λ based on a heuristic derived from Song [36]'s Corollary 2. It tells us that the optimal linear combination of any two estimators $\hat{T}_1$ and $\hat{T}_2$ (when the sum of the weights is one) depends on the biases, variances, and covariance of the two estimators. If the first estimator is unbiased and the variance of the second is zero, we can show that the optimal estimator is $\lambda\hat{T}_1 + (1-\lambda)\hat{T}_2$, where $\lambda = b_2^2/(b_2^2 + v_1)$, b_2 denotes $\hat{T}_2$'s bias, and v_1 denotes $\hat{T}_1$'s variance. To apply this result, we assume that the main factors that might prevent gp-IRT from being a good estimator are the variance of the first estimator and the bias of the second one. Then we approximate the first estimator's bias and the second estimator's variance by zero. When our first estimator is obtained by random sampling we take

$$\lambda = \frac{\hat{b}^2}{\hat{\sigma}^2/|\widehat{\mathcal{I}}_j| + \hat{b}^2}$$

for two constants $\hat{\sigma}^2$ and $\hat{b}^2$. The first constant, $\hat{\sigma}^2$, is obtained by computing the average sample variance of Y_{il}, $i \in \mathcal{I}_j$, across LLMs in the training set. The second constant, $\hat{b}^2$, is obtained by approximating the IRT bias. We (i) split the training set into two subsets of LLMs; (ii) fit an IRT

model in the first part using data from all scenarios; (iii) fit the ability parameter for all the LLMs in the second part using half of the examples of all scenarios; (iv) use that IRT model to predict the correctness (using predicted probabilities) of the unseen examples of scenario j for the models in the second split; (v) average predictions and actual correctness within models, obtaining predicted/actual scenarios scores; (vi) compute their absolute differences, obtaining individual error estimates for models; (vii) average between models, obtaining a final bias estimate, and then square the final number. To give some intuition on how λ is assigned, Figure 3 depicts λ as a function of $\hat{b}$ and $|\widehat{\mathcal{I}}_j|$ when $\hat{\sigma}^2 = .01$. From that figure, we see that if the IRT model bias is small, more weight will be given to p-IRT. The curves are steeper when $|\widehat{\mathcal{I}}_j|$ is small because the variance of the first estimator decreases faster when $|\widehat{\mathcal{I}}_j|$ is small. When the first estimator is obtained by a method that implies an estimator with smaller variance, *e.g.*, anchor points, we apply the same formula but divide $\hat{\sigma}^2$ by a constant > 1. By default, we divide $\hat{\sigma}^2$ by 4 which is equivalent to halving the standard deviation of the first estimator.

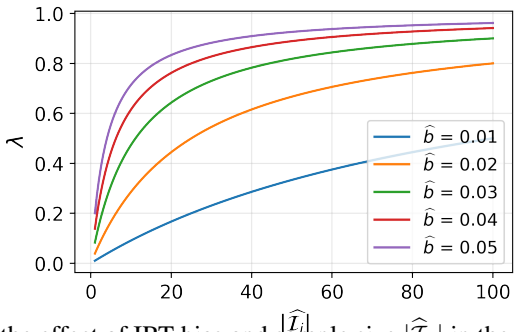

Figure 3: Understanding the effect of IRT bias and sample size $|\widehat{\mathcal{I}}_j|$ in the gp-IRT construction: both quantities are positively related to the weight we give to the raw data in performance estimation.

B.1 Using IRT when Y_{il} is not binary

There are situations in which $Y_{il} \notin \{0, 1\}$ but $Y_{il} \in [0, 1]$. For example, in AlpacaEval 2.0, the response variable is bounded and can be translated to the interval $[0, 1]$. Also, some scenarios of HELM and the Open LLM Leaderboard have scores in $[0, 1]$. We propose a simple and effective fix. The idea behind our method is to binarize Y_{il} by defining a second variable $\tilde{Y}_{il} = \mathbb{1}[Y_{il} \geq c]$, for a scenario-dependent constant c. More concretely, for each scenario j, we choose c such that

$$\sum_{i \in \mathcal{I}_j, l \in \mathcal{L}_{tr}} Y_{il} \approx \sum_{i \in \mathcal{I}_j, l \in \mathcal{L}_{tr}} \mathbb{1}[Y_{il} \geq c].$$

In that way, approximating the average of $\tilde{Y}_{il}$ and Y_{il} should be more or less equivalent. Given that $\tilde{Y}_{il} \in \{0, 1\}$, we can use the standard IRT tools to model it.

B.2 Fitting the IRT model

For the estimation procedure, we resort to variational inference. In particular, we assume that $\theta_l \sim N(\mu_\theta \mathbb{1}_d, 1/u_\theta I_d)$, $\alpha_i \sim N(\mu_\alpha \mathbb{1}_d, 1/u_\alpha I_d)$, and $\beta_i \sim N(\mu_\beta, 1/u_\beta)$. To take advantage of software for fitting hierarchical Bayesian models [21], we introduce (hyper)priors for the prior parameters $\mu_\theta \sim N(0, 10)$, $u_\theta \sim \Gamma(1, 1)$, $\mu_\alpha \sim N(0, 10)$, $u_\alpha \sim \Gamma(1, 1)$, $\mu_\beta \sim N(0, 10)$, and $u_\beta \sim \Gamma(1, 1)$. Finally, to obtain point estimates for the model and example-specific parameters θ_l, α_i, and β_i, we use the means of their variational distributions. To select the dimension of the IRT model during the fitting procedure, we run a simple validation strategy in the training set and choose the dimension that maximizes the prediction power of the IRT model in the validation split–we consider the dimensions in $\{2, 5, 10, 15, 20\}$.

C Details about benchmarks

We describe the size and composition of the four benchmarks, as well as the corresponding LLMs (see Appendix C for additional details):

- HuggingFace's Open LLM Leaderboard [4] consists of 6 scenarios, 29K examples in total. Performance on each of the scenarios is measured with accuracy and the overall benchmark performance is equal to the average of scenario accuracies. We collect evaluation results for 395 LLMs from the Leaderboard's website and use 75% for training and 25% for testing (split either randomly or by date as described above). To extract data from those models, we filter all models from the platform that have an MMLU score over .3, order them according to their average performance, and equally spaced selected models. Then, we kept all models that had scores for all six scenarios: ARC [12], HellaSwag [45], MMLU [15], TruthfulQA [24], Winogrande [35], and GSM8K [13]. In a second round of data collection, we collected data for 40 "specialized models" by recognizing which models were fine-tuned to do the math, coding, *etc.*. The two sets of models have an intersection, and in total, we have collected data from 428 LLMs.

- MMLU [15] is a multiple choice QA scenario consisting of 57 subjects (subscenarios) comprising 14K examples. Performance on MMLU is measured by averaging the accuracies on each of the categories. MMLU is one of the 6 scenarios of the Open LLM Leaderboard and we consider the same set of 395 LLMs and train-test splits. The reason to consider it separately is its immense popularity when comparing LLMs [39, 1, 38] and inclusion into several other benchmarks.

- For HELM [23] we consider the 15 core scenarios (total of 20K examples) and 28 models that have their performances registered for all scenarios as in Perlitz et al. [32]. Performance metrics for each scenario vary and can be non-binary (e.g., ROUGE score), and the overall performance on the benchmark is measured with mean win rate across scenarios. For this benchmark, the dates models were added are not available. Instead, we split models based on the organizations that trained them to create more challenging train-test splits, e.g., all OpenAI models are either in train or in test. For the random train-test split we use 5-fold cross-validation. We average correctness over all repetitions over all trials. The scenarios are BoolQ [11], Civil Comments [6], HellaSwag [45], OpenbookQA [28], IMDB [26], MMLU [15], MS MARCO (Regular track) [31], NarrativeQA [18], NaturalQuestions (closed-book) [19], NaturalQuestions (open-book) [19], QuAC [10], RAFT [2], CNN/DM [29], and XSUM [30].

- AlpacaEval 2.0 [22] consists of 100 LLMs evaluated on 805 examples. Although it is a fairly small benchmark, evaluation is expensive as it requires GPT-4 as a judge. For each input, GPT-4 compares the responses of a candidate LLM and a baseline LLM (currently also GPT-4) and declares a winner. The average win rate[3] is used to measure the overall performance. When splitting the data by date, we pick 25% most recent models for testing and the rest for training. For the random split, we employ 4-fold cross-validation.

D Extra results

D.1 Expanding results in Section 5

Figure 4 is an extension of Figure 2, including random splits in the top row.

[3] AlpacaEval 2.0 considered in the experiments uses continuous preferences instead of binary.

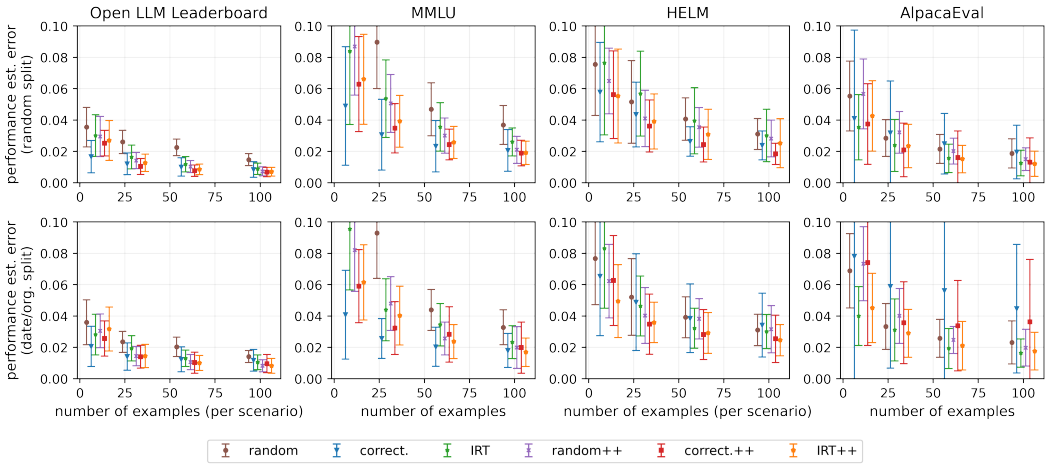

Figure 4: Performance estimation error per benchmark (columns) tested on random (top row) and recent (bottom row) LLMs for increasing number of evaluation examples. 100 examples per scenario is sufficient to achieve ≈2% average performance estimation error across benchmarks and evaluated LLMs. This corresponds to 600 out of 29K examples for Open LLM Leaderboard, 100 out of 14K examples for MMLU, 1500 out of 20K examples for HELM, and 100 out of 800 examples for AlpacaEval 2.0.

Figure **??** is an extension of Figure 1 for more benchmarks and methods.

D.2 Specialized LLMs.

In our previous experiments, the test set of LLMs consisted of either a random subset of models or the most recent ones. Both of these test sets are dominated by base and instruction-tuned LLMs. Here we assess the ability of the considered strategies to predict the performance of specialized LLMs, i.e., models fine-tuned for specific domains such as code, biology, or finance. We consider MMLU benchmark and collect a new hand-picked test set of 40 specialized models. Such models are likely to have unique strengths and perform well in specific MMLU categories while relatively underperforming on others. Thus, their correctness patterns might be different from those in the train set, posing a challenge for our evaluation strategies. We present results in Figure 6.

As we anticipated, the correctness-based anchor strategy deteriorates when tested on specialized LLMs. In contrast to the IRT-based anchors that are only slightly affected, demonstrating their robustness and supporting our choice to use them for tinyBenchmarks construction.

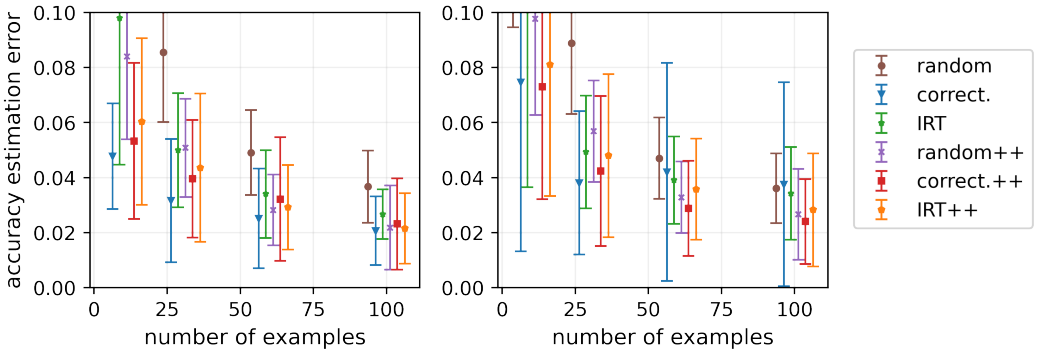

Figure 6: Estimation error on specialized LLMs (right) compared to error on random LLMs (left) on MMLU. Correctness-based example selection is affected the most by this distribution shift.

D.3 ESTIMATION ERROR ANALYSIS

We present a more detailed view of the estimation error of the best performing "IRT++" evaluation strategy on MMLU with 100 examples. In Figure 7 we plot estimation error against the actual accuracy of 99 test LLMs for a random train-test split. Our strategy can estimate the performance of more capable LLMs slightly better, although there is no strong dependency. We also note that the estimation error never exceeds 4% (recall that the average error is 2% as shown in Figure 4), supporting the reliability of our evaluation approach.

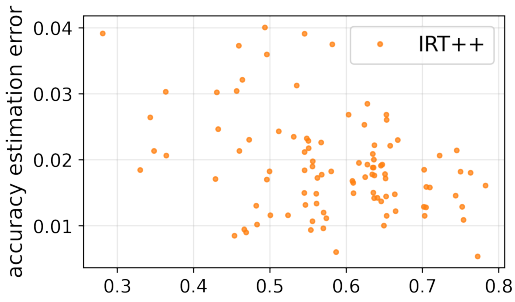

Figure 7: Spread of estimation errors across a random subset of LLMs with varying capabilities on MMLU. Error tends to be slightly lower for more capable models. Worst case error across all models is $\leq 4\%$.

D.4 RUNNING TIME

We record the running time of IRT inference (ability parameter fitting) when running our experiments. Below, we show the average running time depending on the size of used examples (from all scenarios).

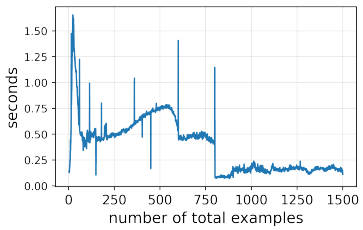

Figure 8: Average running time by the amount of test examples: IRT inference.

D.5 RANK CORRELATION RESULTS

In this section, we explore versions of Figures 4 and 6 when we look at rank correlation (correlation between true and predicted ranking) instead of performance. It is clear from the plots below that our method can be used to rank models efficiently with tiny samples.

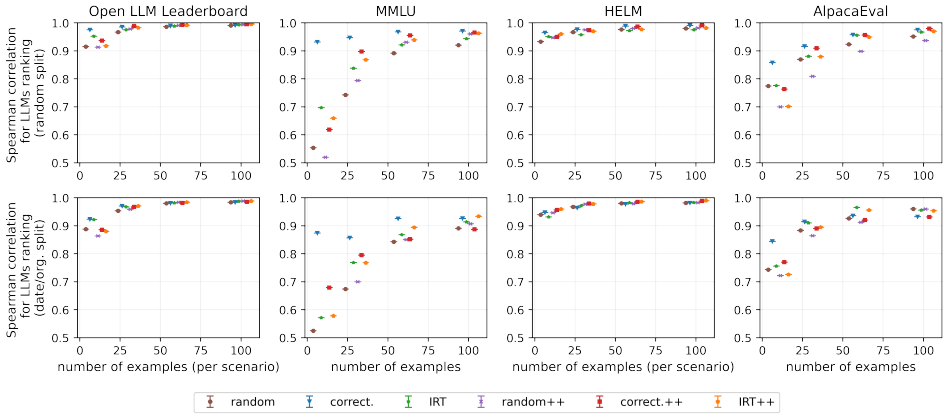

Figure 9: Rank correlation for true performance and predicted performance among LLMs.

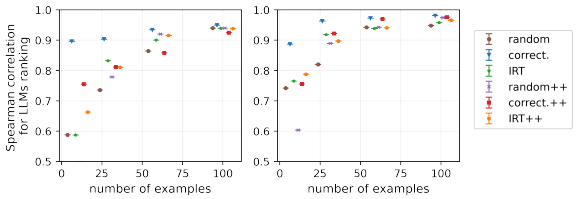

Figure 10: Rank correlation for true performance and predicted performance among LLMs in MMLU. The plot on the left represents a random split of the data while the plot on the right considers specialized models as the test set.

D.6 Adaptive testing

In this section, we complement the results shown in Figure **??** for all benchmarks.

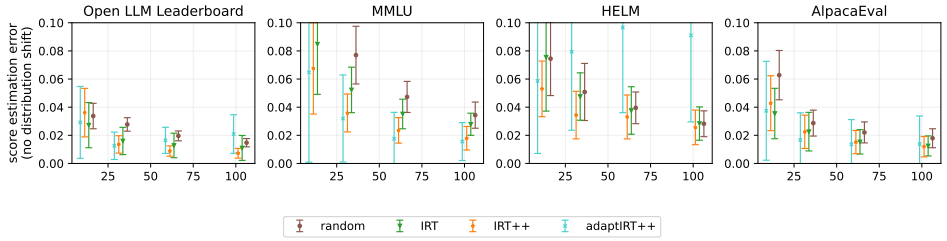

Figure 11: Results of adaptive testing for different benchmarks.

E tinyMMLU

To construct tinyMMLU we chose 100 examples and weights identified by the IRT anchor point approach ("IRT") corresponding to the best test performance (across random seeds) in the experiment presented in the top part of Figure 4 on MMLU. For comparison, we analogously selected 100 examples with the correctness anchor point method.

To better understand the composition of tinyMMLU, in Figure 12 we visualize the distribution of the weights of the selected examples and compare it to the weights of the correctness anchors. Recall that weights are non-negative and sum to 1. If an item has a weight 0.1, for example, that item has a contribution of 10% in the final estimated score. From Figure 12, we can see that tinyMMLU has more uniform weights compared to its correctness-based counterpart. We measure uniformity

through the effective sample size (ESS) of the example weights. ESS, traditionally used in the Monte Carlo and domain adaptation [14, 27] literature, measures weight inequality in a way such that ESS $= 0.50$, for example, informally means that the corresponding weighted average is influenced by only 50% of (uniformly weighted) examples. In the context of our problem, more uniform weights of tinyMMLU contribute to its robustness when evaluating LLMs with varying correctness patterns, such as specialized LLMs in Figure 6.

We also investigate the total weight of the tinyMMLU examples within each of the 57 subjects in Figure 13. The highest weighted are "high school psychology", "elementary mathematics", and "professional law". Interestingly the weight of the subjects is fairly different from its correctness-based counterpart.

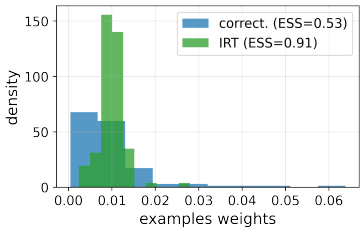

Figure 12: Comparing the spread of examples weights using both the IRT and correctness approaches to find anchor points. We see that weights inequality is much higher when we cluster examples using correctness.

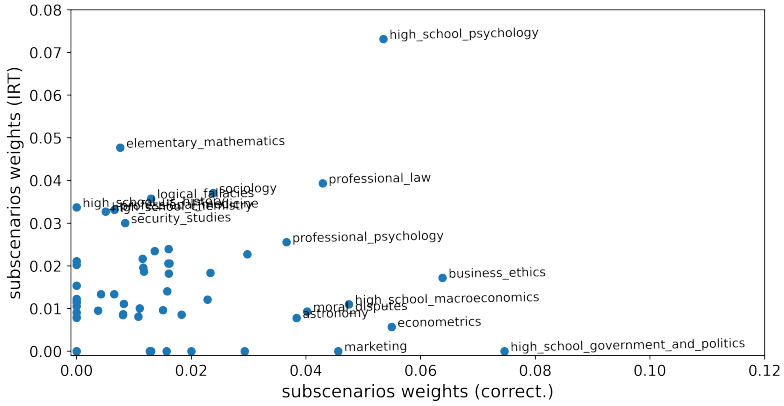

Figure 13: Weights given to MMLU subscenarios by the two anchoring methods.

F INDIVIDUAL PERFORMANCES PER SCENARIO

In this section, we explore what is behind Figure 4 by looking in detail at results for individual scenarios for the Open LLM Leaderboard and HELM. It is clear from the following plots that there are scenarios in which our methods shine more than others.

F.1 OPEN LLM LEADERBOARD

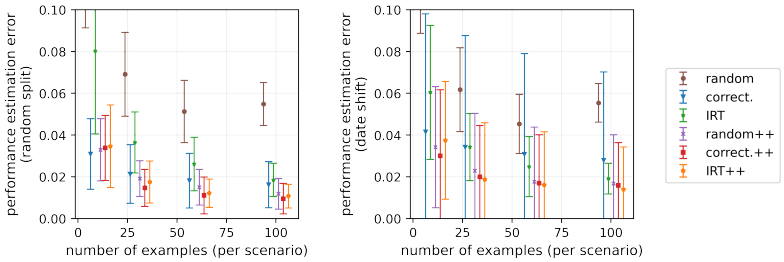

Figure 14: ARC

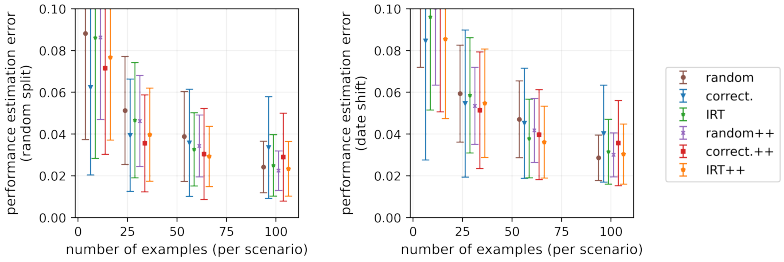

Figure 15: GSM8K

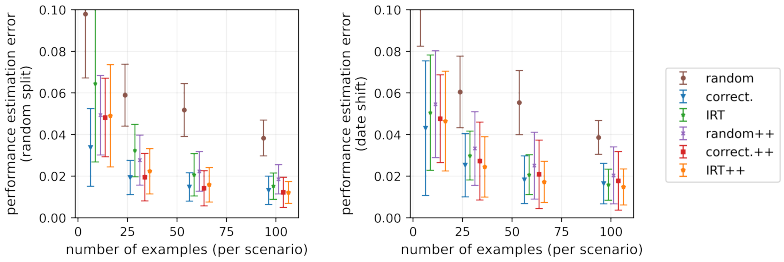

Figure 16: TruthfulQA

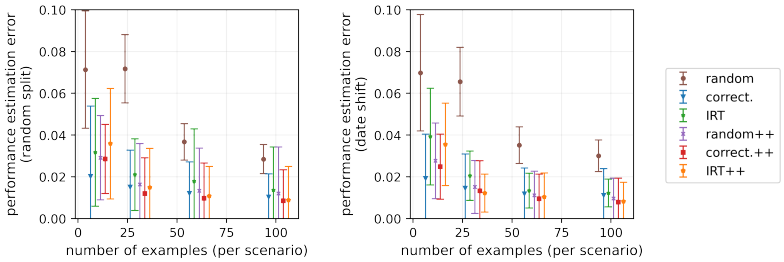

Figure 17: HellaSwag

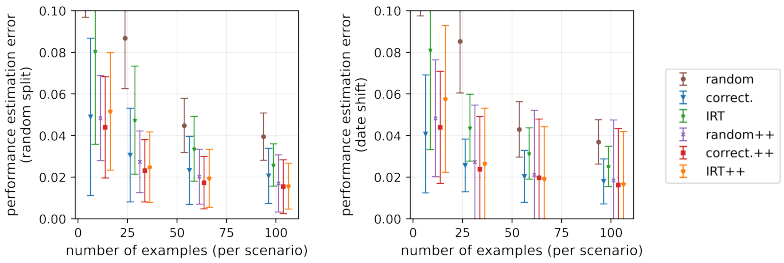

Figure 18: MMLU

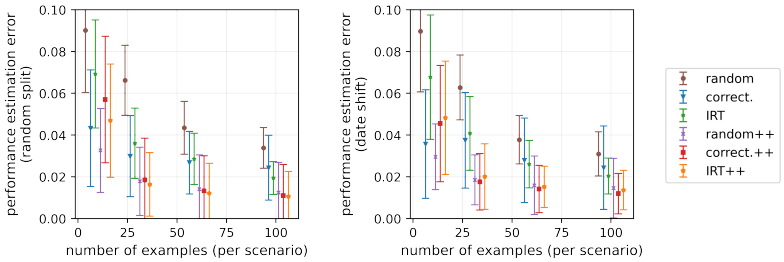

Figure 19: Winogrande

F.2 HELM

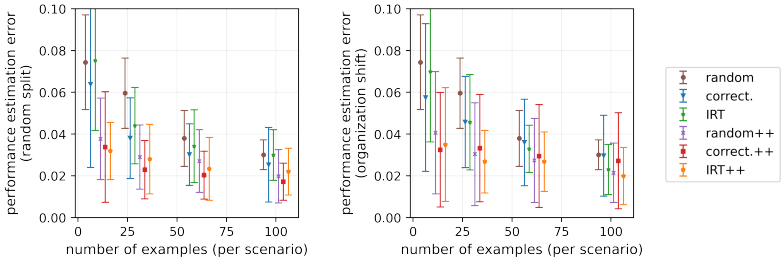

Figure 20: BoolQ

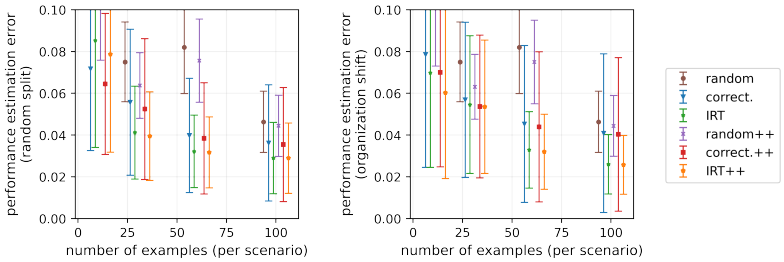

Figure 21: Civil Comments

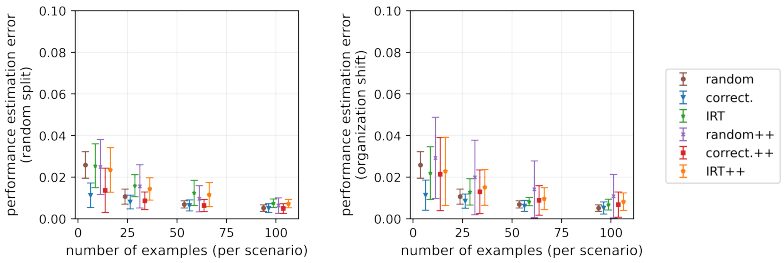

Figure 22: CNN/DM

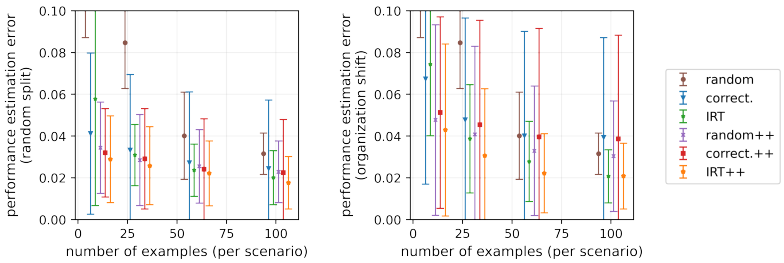

Figure 23: HellaSwag

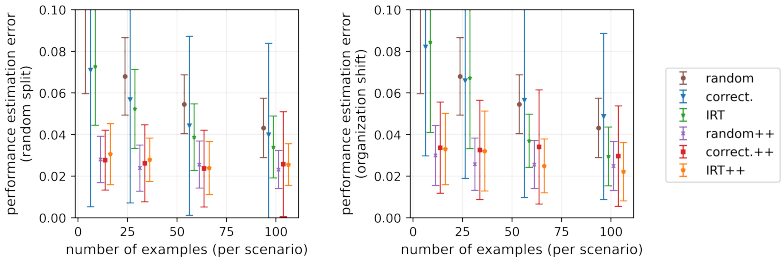

Figure 24: OpenbookQA

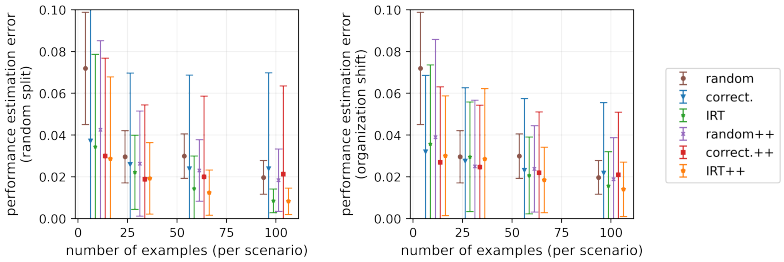

Figure 25: IMDB

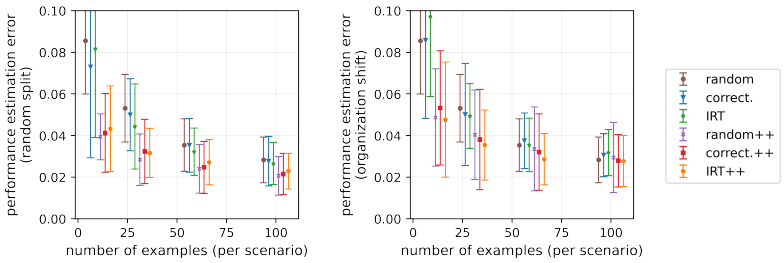

Figure 26: MMLU

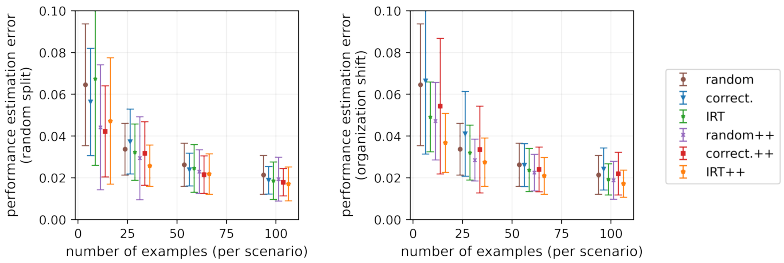

Figure 27: MS MARCO (Regular Track)

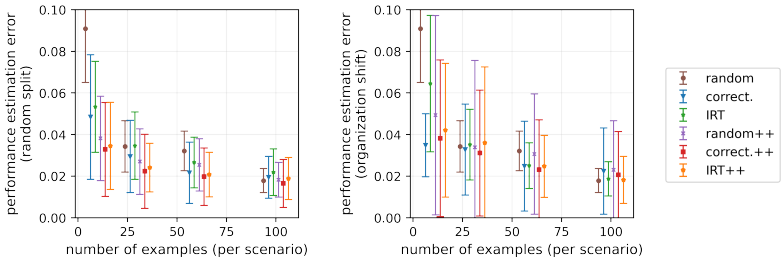

Figure 28: NarrativeQA

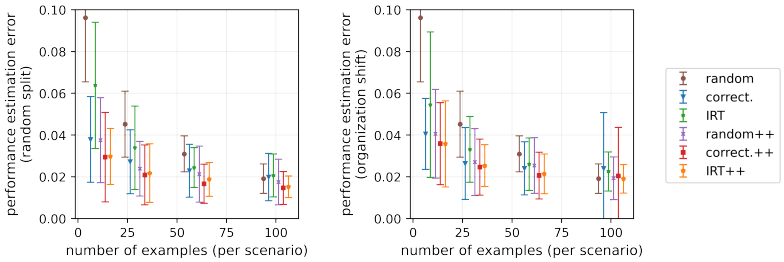

Figure 29: NaturalQA (closed book)

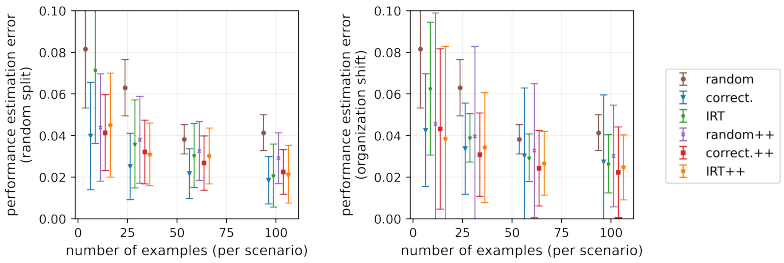

Figure 30: NaturalQA (closed book)

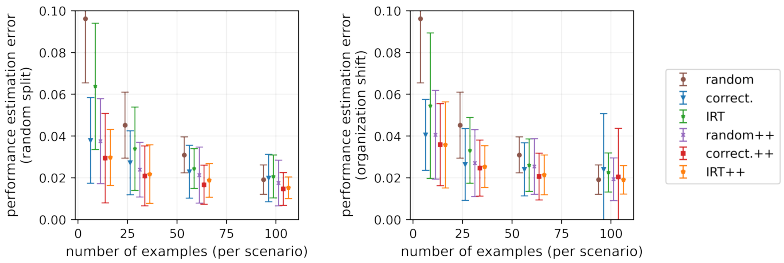

Figure 31: NaturalQA (closed book)

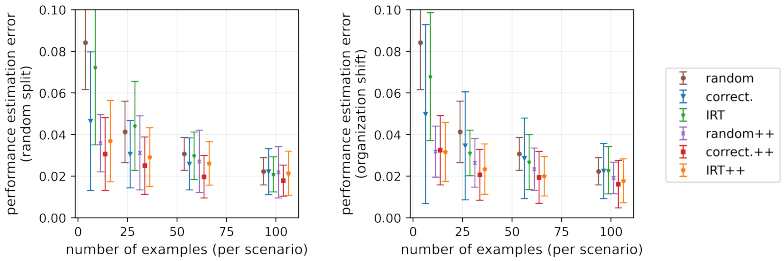

Figure 32: QuAC

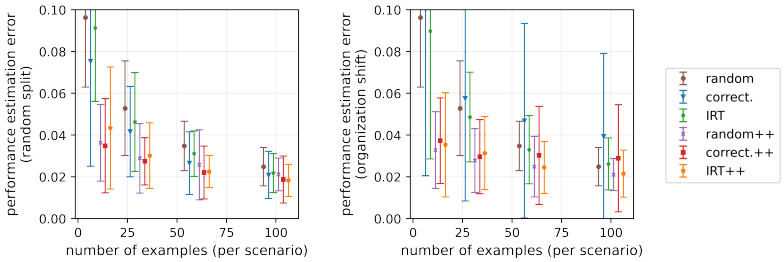

Figure 33: RAFT

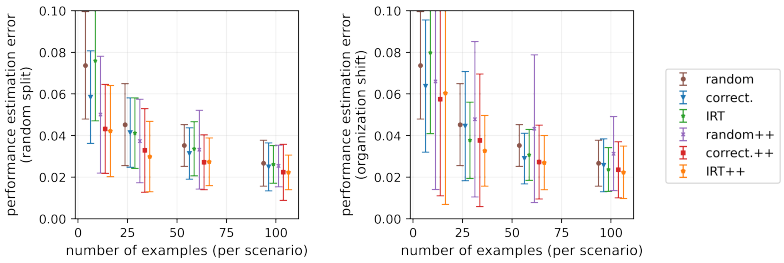

Figure 34: TruthfulQA

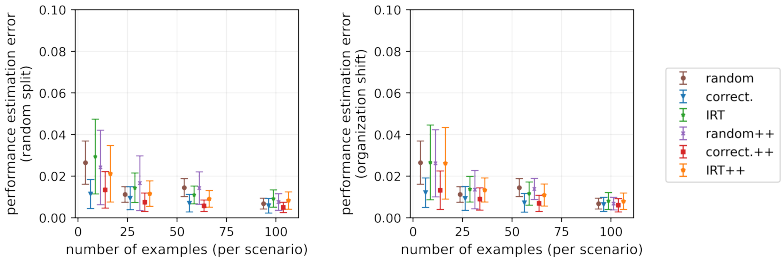

Figure 35: XSUM