# From Understanding to Generation: An Efficient Shortcut for Evaluating Language Models

**Viktor Hangya** and **Fabian Küch** and **Darina Gold**
Fraunhofer IIS
{first.last}@iis.fraunhofer.de

## Abstract

Iterative evaluation of LLMs during training is essential to ensure expected capability development, but can be time- and compute-intensive. While NLU tasks, where the model selects from fixed answer choices, are cheap to evaluate, essential capabilities like reasoning and code generation rely on the more time-consuming NLG (token-by-token generation) format. In this work, our aim is to decrease the computational burden of NLG benchmarks in order to enable monitoring crucial LLM capabilities during model training. We reformulate generative tasks into computationally cheaper NLU alternatives. We test the performance correlation between the original and reformulated tasks using 8 LMs of various sizes and 4 capabilities: mathematical reasoning, code generation, factual knowledge and reading comprehension. Our results show a strong correlation between task formats, supporting capability assessment via cheaper alternatives and achieving over 35× average reduction in evaluation time. Our project is available at: https://github.com/Fraunhofer-IIS/EvalShortcut

## 1 Introduction

The increasing adoption of large language models (LLMs) has brought a substantial rise in computational demands—not only for training, but also for evaluating the performance of new models. Numerous works have explored methods for selecting the best quality training datasets and hyper-parameters based on smaller proxy models, evaluating them multiple times throughout training (Li et al., 2024; Grattafiori et al., 2024; Magnusson et al., 2025). However, due to model capability and compute limitations they rely on a few benchmarks which support measuring a limited set of model capabilities (Penedo et al., 2024; Gu et al., 2025).

Evaluation benchmarks can be grouped into two types: 1) *Natural Language Generation (NLG)* which requires auto-regressive sampling from the model, and 2) *Natural Language Understanding (NLU)* which involves calculating the probability of given answer options (Guo et al., 2023; Biderman et al., 2024). Assessing a model's NLG abilities is costly, especially when done repeatedly during model training and selection, and thus is often neglected (Zhou et al., 2022). Yet, many key capabilities, e.g. reasoning and code generation, are only measurable via NLG benchmarks.

Here, we argue for the importance of monitoring the trajectory of LLM capabilities throughout the (pre-)training process, which is currently reliant on expensive NLG benchmarks.

Our goal is to reduce compute requirements for such benchmarks in order to enable a more frequent use. Towards this end, we propose reformulating expensive NLG benchmarks as less-resource intensive NLU tasks. Specifically, we reformulate it to multiple choice (MC) tasks, where the model has to select the correct answer among answer options, and log-likelihood (LL) tasks where the model calculates the LL of the correct answer. The cost-reduction of MC and LL over then generative task is shown in Table 1. The generative setup completes the response token-by-token. The MC setup, however, evaluates the model's ability to select the appropriate response from a set of alternatives based on their likelihoods, while the LL approach scores only the correct answer. Despite differing formats, all methods aim to assess the model's response to the same underlying task.

Since many NLG benchmarks do not contain incorrect answer options to questions, we create such options using either LLMs or simply picking answers of other samples in a given benchmark randomly. In contrast, LL is simpler to apply, as it evaluates the likelihood assigned by the model to a given correct answer without the need of an overhead, i.e., the computation of alternative answers. As such, it offers further reductions in evaluation costs, however, at the expense of evaluation depth,



*Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 22577–22593

| | Step | <Q> := What is gravity? | | Step output | Final output |
|---|---|---|---|---|---|
| LL | 1. | <Q> Downward force on objects. | → | $log(0.9)$ | Log-likelihood of correct answer: $log(0.9)$ |
| MC | 1. | <Q> Downward force on objects. | → | $log(0.9)$ | Selected answer index: 1. |
| | 2. | <Q> What makes the sun rise. | → | $log(0.1)$ | |
| NLG | 1. | <Q> | → | Downward | Generated answer: Downward force on objects. |
| | 2. | <Q> **Downward** | → | force | |
| | 3. | <Q> Downward **force** | → | on | |
| | 4. | <Q> Downward force **on** | → | objects. | |
| | 5. | <Q> Downward force on **objects.** | → | [EOS] | |

Table 1: Comparison of answers to the same question in the natural language understanding variants (NLU) —loglikelihood (LL) and multiple-choice (MC)—and natural language generation (NLG) settings. LL scores the correct answer to the question (Q), MC selects the most probable answer option, while NLG generates the answer token-by-token. The column *step* indicates the number of forward model passes needed for the answer calculation. The example and outputs (e.g., log values) are fictional and serve solely to illustrate the NLU–NLG computation differences.

since high-probability incorrect answers are not considered. MC and LL evaluation allows quick comparison of model checkpoints and low-cost training monitoring. Importantly, our goal is not to replace the NLG format, but to offer a cheaper alternative for monitoring model capabilities during pre-training.

To test this hypothesis, we conduct a study involving a diverse set of benchmarks representing various domains. We analyze the correlation of 8 open source model performances between the NLG and NLU task formulations, as well as the correlation of model rankings. To our knowledge, this is the first study to systematically investigate this relationship. We find a strong correlation, indicating the ability to monitor the performance trajectory when training a single model, but also the ability to compare the model in question to baseline LLMs using only the NLU formulated benchmarks. Additionally, we conduct ablation studies, e.g., pairing NLG and NLU benchmarks of the same domain but different sources, finding promising results in these settings, too. On top of the correlation experiments, we show that besides monitoring the model development trajectory using NLU tasks, it is also possible to predict the NLG performance of a model by fitting a linear regressor and only occasional NLG evaluations. This optional step balances precise NLG performance monitoring and compute resource needs.

Our main contributions are threefold:

- we propose and apply a methodology for reformulating NLG tasks as NLU tasks for more efficient benchmark evaluation and broader capability assessment during model selection and training
- we show strong correlation between task formulations and analyze various aspects; a very important one being that by reformulating NLG tasks we achieve a compute time reduction of 35x on average, reducing runtime from nearly 2 hours to under a minute in the most extreme case.
- we publicly release our evaluation framework.

## 2 Related Work

This section is divided into three subsections, each addressing key components of this work. In Section 2.1, we discuss evaluation costs and efforts to reduce them. In Section 2.2, we review studies on the relation between NLG and NLU evaluation. In Section 2.3, we present knowledge categories used to study LLM generalizability.

### 2.1 Evaluation costs

Various benchmarks, including NLG with many samples, have been proposed to evaluate LLMs, but evaluating large models on them can be costly. For example, Liang et al. (2023) introduce the HELM benchmark which could cost over 4K GPU hours to evaluate a single LLM, not to mention monitoring the performance over the course of the training process (Biderman et al., 2023; Liu et al., 2024). Recent studies have proposed various strategies to address reducing the evaluation costs. While Perlitz et al. (2024) and Polo et al. (2024) reduce the number of evaluation items using subsampling, Kuramoto and Suzuki (2025) predict fine-tuning outcomes on large datasets using results from smaller-scale experiments. Compressed benchmarks may reduce evaluation diversity and underestimate model limitations by missing out on key evaluation samples. While Polo et al. (2024) retain key samples via an information-theoretic approach, their approach requires sample-level correctness results of a set of LLMs making its application difficult. In contrast, we reformulate NLG tasks to NLU independently from the target LLM, reducing costs while retaining all samples.

### 2.2 Linking NLG and NLU

Several studies attempt to use NLU tasks to assess NLG and reduce costs (Zhang et al., 2024; Khashabi et al., 2020; Li et al., 2023; Myrzakhan et al., 2024), but mainly target single tasks, such as mathematical reasoning (Zhang et al., 2024) or question answering (QA) (Khashabi et al., 2020).

Most similar to our work, Zhang et al. (2024) compare MC and NLG versions of math and coding benchmarks, showing evaluation time can be reduced by up to 30×. Our work however, differs in multiple aspects, such as we explore both MC and LL reformulations, we explore and use multiple approaches to build answer options for MC tasks including a lightweight method which does not rely on LLM-based generation, we test our approach on a diverse set of capabilities and present the first systematic analysis of NLG–MC benchmark pairings showing consistently positive correlation. Furthermore, we show positive correlation on all of them in contrast to Zhang et al. (2024) who showed mixed results.

Khashabi et al. (2020) introduce a single QA model that performs well across multiple formats, including MC and generative. Their study shows that knowledge learned from MC tasks transfers effectively to generative QA. However, their approach requires model adaptations, while our approach requires reformulating a given benchmark once which is then applicable to any LLM.

Li et al. (2023) propose a method to create MC benchmarks for evaluating multimodal LLMs. They do so by prompting language models with visual content extracted from images or videos to generate one question and four answer options (one correct), effectively converting open-ended understanding into a structured MC format. We follow a similar approach, to generate distractor answer options in our MC reformulations.

Finally, Myrzakhan et al. (2024) convert MC benchmarks into open-ended formats to better assess generative abilities and reduce biases such as guessing. They find that open-style questions score lower, suggesting MC questions may overestimate model ability. While Myrzakhan et al. (2024) did not report the correlation between MC and NLG formats, we computed Pearson correlation showing a positive relation between task formulations.

### 2.3 Task categorization for LLM evaluation

As noted, prior NLG–NLU studies focus on specific tasks; we aim to generalize across multiple capabilities. Wang et al. (2024b) evaluate LLMs' reliability in MC questions across knowledge, language, understanding, and reasoning tasks. Wang et al. (2024a) break down evaluation in four different neural capabilities: linguistic knowledge, formal knowledge, world modeling, and social modeling. In this work we consider four LLM capabilities: mathematical reasoning, factual knowledge, reading comprehension and code generation.

## 3 Experiment Setup

In this section, we describe our experiment setup, its components and motivate our choices. This paper investigates the hypothesis that LL and MC formulations can serve as effective proxies for assessing generative model capability development over the course of training, while significantly reducing the computational cost of evaluation. To evaluate this hypothesis, we pair different formulations of the same or related benchmarks, analyze the correlation between the NLG and NLU versions, and compare their evaluation runtimes.

### 3.1 Evaluation variants

We use three different variants of each benchmark: the original NLG version, the MC version and the LL version. In the NLG variant, the model receives a question and must produce an answer as free text. In the LL and MC variants, it is given the question along with either a single correct answer option (LL) or multiple (a correct and multiple incorrect) answer options (MC) and calculates the probabilities assigned to them. In Table 1, we illustrate these settings.

To judge the accuracy of the outputs different metrics are considered depending on the task. For the generative formulation, exact (token-by-token) or proximity (e.g. BLEU) matching with the gold answer are used. For the MC formulation, the model is considered to have selected the correct answer if it was the one with the highest probability. Since LL only scores the correct answers, we calculate the average log-probability ($mean_{i=1..N} log \hat{P}(y_i|x_i)^1$) of correct answers ($y_i$)

[1]We take (character) length normalized values:

of questions ($x_i$) in a given benchmark and we expect that it correlates with quality of the model's capability without calculating an exact correctness value. We detail calculating correlation in Section 3.2.

As shown in Equation 1, log-likelihood evaluates the probability of a fixed target sequence in a single forward step, multiple choice evaluates the given ($K$) candidate sequences in one forward step each, while generative decoding generates one token in each forward step over $T$ time steps (maximum output tokens). For further details about the exact calculations we refer to Gao et al. (2024).

$$\text{Compute Complexity} = \begin{cases} \mathcal{O}(1) & \text{(LL)} \\ \mathcal{O}(K) & \text{(MC)} \\ \mathcal{O}(T) & \text{(NLG)} \end{cases} \quad \text{where } K \ll T \tag{1}$$

### 3.2 Metrics

To evaluate how indicative NLU task formulations are on the generative performance, we use both intra- and cross-model metrics.

The goal of intra-model metrics is to measure how similarly the performance of NLU and NLG versions of the same task change over the training course of a model. See Figure 1 for a visualization. We calculate Pearson correlation between the NLU and NLG results of the intermediate model checkpoints over the course of training. We define $P_{macro}$ and $P_{micro}$ as macro- and micro-averaged Pearson correlation across models respectively, i.e., calculating the correlation values for each of our 8 considered models separately followed by averaging the model-wise correlation values, and calculating the correlation using data points of all models jointly. High correlation values indicate that improvements or decrease of model performance on an NLU task during the training process means performance change in the same direction in the NLG format as well.

In contrast, the goal of cross-model metrics is to measure the consistency of the ranking of our 8 models in the different task formulations. For this, we take the model rankings based on averaged performance over all intermediate checkpoints per model and calculate Spearman correlation. High coefficients indicate the possibility of comparing the model of interest to other models using the NLU task formulation instead of running the costly NLG version.

$log\hat{P}(y_i|x_i) = logP(y_i|x_i)/|y_i|$.

### 3.3 Benchmarks

To show the broad applicability of our approach, we use four distinct knowledge domains. This section discusses existing benchmarks; benchmark reformulations are detailed in Section 3.4. Examples for all benchmarks discussed herein as well as their extensions are shown in Table 9 in the appendix.

**NLG Benchmarks** We leverage GSM8K (Cobbe et al., 2021) for mathematical reasoning, HumanEvalPack (Muennighoff et al., 2023)[2] for code generation (referenced as HumanEval further on), TriviaQA (Joshi et al., 2017) for factual knowledge and SQuAD 2.0[3] (Rajpurkar et al., 2018) for reading comprehension.

**NLU Benchmarks** We discuss off-the-shelf NLU benchmarks in this section, while the reformulation of NLG tasks as MC is covered in Secion 3.4. Zhang et al. (2024) created a MC version of GSM8K by using incorrect answers of LLMs, and only the numeric value (see Table 9), to a given question as MC options. We use this version of GSM8K, besides our own, to represent MC tasks for mathematical reasoning. Additionally, we run ablation studies where we test cross-benchmark pairings, see Section 3.4. For these experiments, we use MMLU (Hendrycks et al., 2021) for factual knowledge and BoolQ (Clark et al., 2019) for reading comprehension.

### 3.4 Multiple choice distractor creation

Since most NLG benchmarks only provide the correct answers, which is already enough for LL, we had to create incorrect answer options in order to reformulate them as MC tasks. We focus on the creation of MC benchmark variants using existing NLG benchmarks, as done by Zhang et al. (2024). We created both random and smart negative answer options (3 if not stated otherwise) which we discuss next. Table 2 shows exemplary random and smart distractors to the same question.

**Random distractors** For GSM8K, TriviaQA, and HumanEvalPack, we created random distractors[4] by using answers of other questions within the same benchmark. For SQuAD, we did it similarly, but used only answers of questions that have

[2] This is the extension of HumanEval (Chen et al., 2021) from Python to 5 more code languages: C++, Go, Java, JavaScript and Rust

[3] As we use only this version of *SQuAD*, it will be referenced without the version further on.

[4] We only test a single random seed for efficiency reasons.

| | |
|---|---|
| Question | Which was the first European country to abolish capital punishment? |
| Correct answer | Norway |
| Random distractors | [Chicago Bears, Ballet, 6] |
| Smart distractors | [Germany, Italy, Poland] |

Table 2: Example for random and smart distractors in TriviaQA.

the same context as the question at hand.[5] This ensured that they were formally valid, but semantically incorrect. However, we hypothesized that models might disregard the incorrect answers not because they knew the correct one, but because the incorrect options were implausible or clearly irrelevant to the question, e.g. as shown in Table 2, the question is looking for a *European country* and none of the random distractors is a location. Hence, we also created *smart distractors*.

**Smart distractors** To generate three plausible incorrect answers—*smart distractors*, we prompted a Meta-Llama-3.1-70B-Instruct-GPTQ-INT4 model (using a benchmark-specific 5-shot prompt; see Appendix A) for two NLG benchmarks, namely TriviaQA and GSM8K. Table 2 illustrates how smart distractors, unlike *random distractors*, match the semantic context (e.g. European countries). To assess the plausibility of the automatically generated smart distractors, one author manually reviewed 50 random question samples and their corresponding distractors for each benchmark. Out of the overall 100 questions, one distractor was a correct answer, and in eight cases, two of the distractors were repeated, leaving only two distinct wrong answers. After we developed our initial prompt for the first task (Appendix A), adapting it to the next took under 30 minutes. Depending on the task, running the distractor generation took between 16-24 hours, including some manual checks and fixing JSON parsing errors due to incorrect LLM outputs. We believe this demonstrates that our prompt-based approach is an effective and fast method for generating smart distractors. For SQuAD, we omitted smart distractors since random distractors already share context with the question. For HumanEvalPack, as smart distractors we use the slightly altered incorrect code snippets provided by the dataset for the code debugging task, resulting in two MC options instead of four.

**Cross-benchmark pairing** On top of converting NLG benchmarks to MC using either random or smart distractor generation, we experiment with pairing NLG benchmarks with off-the-shelf MC datasets targeting similar capabilities. However, we note that these secondary experiments serve only as a scientific exploration to test performance correlation on related but different datasets, since the above mentioned distractor generation is trivial and not all LLM capabilities have off-the-shelf NLG and MC benchmarks available. To the best of our knowledge, this is the first work to conduct such a pairing analysis between NLG and MC benchmarks. We found pairings for two of our NLG datasets. To represent *factual knowledge*, we use MMLU as the MC version of TriviaQA. Similarly, to represent *reading comprehension*, we use BoolQ as the MC version of SQuAD.

### 3.5 Models

We selected open base models of varying sizes and training stages, enabling evaluation across a range of performance levels and intermediate checkpoints, which aligns with our motivation to efficiently monitor LLM capabilities during pretraining. We use the Pythia[6] 1B, 2.8B and 6.9B (Biderman et al., 2023), Amber 7B (Liu et al., 2024), OLMo 1B and 7B (Groeneveld et al., 2024), and OLMo-2 7B models (OLMo et al., 2024), as well as the code-specific Crystal 7B model (Liu et al., 2024). We did not include larger LLMs due to compute restrictions. We evaluated intermediate model checkpoints at 20B, 40B, 60B, 100B, 300B, 1T, 1.3T, 2T, 3T and 4T tokens. Note that not all models were trained up to 4T tokens. Please see Table 8 for more details. We present results based on the amount of FLOPS used to produce a given checkpoint in figures 1 and 2. We follow the standard approach to estimate FLOPS based on model parameter[7] (N) and token (D) counts: $FLOPS = 6ND$ (Kaplan et al., 2020).

[5] SQuAD questions are created based on Wikipedia articles. Each article yields multiple questions. Table 9 shows two questions based on the same article.

[6] We used the deduplicated version.

[7] We exclude embedding parameters as suggested by (Kaplan et al., 2020).

| NLG | NLU | $P_{macro}$ | $P_{micro}$ | Spearman |
|---|---|---|---|---|
| GSM8K | MC | 0.75 (0.12) | 0.52 (0.00) | 0.76 (0.03) |
| | $MC_{rnd}$ | 0.76 (0.11) | 0.57 (0.00) | 0.76 (0.03) |
| | LL | 0.79 (0.09) | 0.56 (0.00) | 0.81 (0.01) |
| Trivia | MC | 0.90 (0.03) | 0.94 (0.00) | 0.86 (0.01) |
| | $MC_{rnd}$ | 0.91 (0.02) | 0.88 (0.00) | 0.98 (0.00) |
| | LL | 0.90 (0.03) | 0.69 (0.00) | 0.81 (0.01) |
| SQuAD | $MC_{rnd}$ | 0.90 (0.03) | 0.88 (0.00) | 0.93 (0.00) |
| | LL | 0.65 (0.15) | 0.85 (0.00) | 0.69 (0.06) |
| Human Eval | MC | 0.83 (0.09) | 0.79 (0.00) | 0.81 (0.02) |
| | $MC_{rnd}$ | 0.85 (0.07) | 0.75 (0.00) | 0.81 (0.02) |
| | LL | 0.86 (0.07) | 0.73 (0.00) | 0.79 (0.03) |

Table 3: Correlation statistics of NLG tasks and their various reformulated formats. We present p-values in parentheses. HumanEval results averaged over the 6 coding languages. MC stands for the multiple-choice with smart distractors, $MC_{rnd}$ random distractors, LL is the log-likelihood formulation.

### 3.6 Technical details

For our experiments, we used the *lm-eval-harness* (Gao et al., 2024), due to its wide adoption, extensibility, and support for a broad range of benchmarks. We extend this framework by adding setups of our new LL and MC formulations of the mentioned benchmarks. The only exception is HumanEval for which we use the *bigcode-evaluation-harness* (Ben Allal et al., 2022) as it supports safe code execution. Still, we add its NLU formulations to the lm-eval-harness as these formats do not need code execution. We use default parameters of the evaluation frameworks, except that we use a 0-shot setup for all tasks other than all variants of GSM8K, for which we use 5-shot examples. We follow the suggestions of OLMES (Gu et al., 2024) when evaluating MC benchmarks, i.e., we use the completion (cloze) formatting, where models score answer options instead of answer labels (A, B, C, D, etc.), and we use length normalized probability values to select the final answer for accuracy calculation. For further details we refer to the above mentioned papers.

## 4 Results

Table 3 shows our main results of the four considered task categories. Each row shows the correlation coefficients for the results of the given task in the NLG and indicated reformulated setting. We present averaged code results on 6 coding languages (cpp, go, java, js, python and rust).[8] As introduced, we have three reformulated versions of the NLG tasks: MC and $MC_{rnd}$ indicate results for the multiple-choice reformulation with smart and random distractors respectively, while LL represents the log-likelihood reformulation. Additionally, Figure 1 shows task performance curves over the course of training, averaged across models and standardized for each task formulation to bring different formulations to more visually comparable scales.

Overall, Table 3 demonstrates a strong correlation between the NLG and both MC task variants, as well as between NLG and LL, in all three metrics. This trend is also supported in Figure 1, showing similar performance shifts on the x-axis between the NLG and NLU formulations. These results suggest that the NLU reformulated tasks are effective indicators during model training for the generative performance and can reduce compute.

Interestingly, MC and $MC_{rnd}$ perform on par with each other. We compared the two formulations on small (1B and 2.8B) and large (6.9B and 7B) models, and found that in case of small models $MC_{rnd}$ performs clearly better than MC ($P_{macro}$: +0.05, $P_{micro}$: +0.14, Spearman: +0.52), while in case of large models we did not find a clear difference ($P_{macro}$: -0.02, $P_{micro}$: 0.00, Spearman: 0.13 when comparing $MC_{rnd}$ to MC). Our conjecture is that less capable models tend to output answers unrelated to the question, e.g., completely wrong GSM8K reasoning sequence, thus can be misled by random answer options more easily, while better quality models understand the context of questions more precisely, thus are less sensitive to random distractors.

Although LL performs on par with MC and $MC_{rnd}$, the correlation values are slightly lower on average, especially on TriviaQA, SQuAD and HumanEval. On the contrary LL performs slightly better on GSM8K. These results are in line with the findings of Schaeffer et al. (2024), who found that it is important to consider the probabilities of a few negative answer options when predicting scaling behavior of LLMs. In addition, our results show that considering negative options in the form of the MC task formulation is beneficial also for monitoring the generative performance of a given model. Although the effort of creating smart and random distractors for the MC formulations is minimal, one can sacrifice a slight performance for the simplicity of LL which only needs the correct answer options. The better performance of LL on GSM8K

[8] Detailed results can be found in Table 7.

| NLG | NLU | $P_{macro}$ | $P_{micro}$ | Spearman |
|---|---|---|---|---|
| GSM8K | $\text{MC}_{rnd}$ | 0.76(0.11) | 0.57(0.00) | 0.76(0.03) |
| | $\text{MC}_{ao}$ | 0.38(0.26) | 0.90(0.00) | 0.88(0.00) |
| | LL | 0.79(0.09) | 0.56(0.00) | 0.81(0.01) |
| | $\text{LL}_{ao}$ | 0.22(0.38) | -0.04(0.80) | -0.07(0.87) |
| Trivia | $\text{MC}_{rnd}$ | 0.91(0.02) | 0.88(0.00) | 0.98(0.00) |
| | MMLU | 0.89(0.04) | 0.94(0.00) | 0.95(0.00) |
| SQuAD | $\text{MC}_{rnd}$ | 0.90(0.03) | 0.88(0.00) | 0.93(0.00) |
| | $\text{MC}_{rnd^*}$ | 0.90(0.03) | 0.84(0.00) | 0.93(0.00) |
| | BoolQ | 0.78(0.11) | 0.24(0.09) | 0.45(0.26) |

Table 4: Correlation statistics of various additional tests. For GSM8K we tested setups where only the final answer has to be scored by the model ($*_{ao}$) as proposed by (Zhang et al., 2024). In case of TriviaQA and SQuAD, we tested cross-benchmark pairings: MMLU and BoolQ respectively. Additionally, in $\text{MC}_{rnd^*}$ we used more than 4 answer options for SQuAD.

is likely attributable to the chain-of-thought (CoT) reasoning steps in the outputs which makes the log-likelihood calculation more reliable. As discussed in section 4.1, when we omit the reasoning steps from the scored output the performance drops significantly. This could indicate the need for more difficult distractor options.

As discussed before, the goal of cross-model metric (Spearman) is to compare different models using the cheaper reformulated tasks. In contrast, the goal of intra-model metrics ($P_{macro}$ and $P_{micro}$) is to show whether reformulated tasks can be used to monitor the NLG performance of a single model during training. We found similarly strong correlation values for both categories, showing their usefulness in both scenarios. Comparing $P_{macro}$ and $P_{micro}$, we found stronger coefficients for $P_{macro}$ indicating that the scaling factor between the NLG and the reformulated task performances slightly varies from model to model.

### 4.1 Ablations

In addition to the above experiments, we evaluated further task formulations to have a better understanding of the important factors which we present in Table 4. For GSM8K, as discussed above, having CoT reasoning steps in the scored outputs is an important factor for the correlation values. Based on the work of Zhang et al. (2024), we evaluate the MC and LL formulations of GSM8K which only include the final numeric answer only in the output ($\text{MC}_{ao}$ and $\text{LL}_{ao}$ respectively). The reasoning steps are crucial for the LL formulation as the correlation coefficients of all three metrics drop

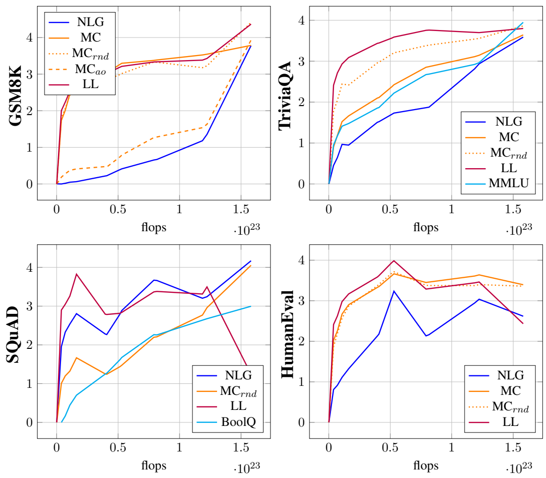


Figure 1: Average performance across models per task formulation at fixed compute (flops). Task formulation performances are standardized to bring different formulations to a more visually comparable scale. Our expectation is that NLG and its reformulations have similar developments over time, i.e., high Pearson correlation.

significantly when removing them. In contrast, providing only the final answer has the opposite effect in case of MC. Figure 1 shows that in contrast to MC, $\text{MC}_{ao}$ reflects the rapid performance increase at later stages more precisely, while MC seems too easy for the models, i.e, the performance rapidly increases at the early training stages and slows down later on. This indicates that harder distractors might be needed for this task.

For TriviaQA and SQuAD we tested cross-benchmark pairing by leveraging the MMLU and BoolQ datasets. Surprisingly, we found strong correlation values using MMLU, even outperforming our $\text{MC}_{rnd}$ formatted TriviaQA version. A possible explanation for this is the longer answer options in MMLU compared to the short options in TriviaQA which could make MMLU more reliable in the MC format. In contrast, BoolQ does not perform as well as SQuAD $\text{MC}_{rnd}$, although it still has a decent correlation ($P_{macro}$). As can be seen in Figure 1, BoolQ nicely reflects the NLG performance increase of SQuAD on average, however, it does not follow the curve as closely. This aligns well with the findings with MMLU, as BoolQ has only short *Yes* and *No* answer options which could make it less reliable. These findings also align with Xiao et al. (2024), who argue for loss calculation on longer outputs when finding the effective parameter size of LLMs, and opt for LLM based CoT generation in case of tasks with short outputs. We leave this for future work, as reasoning generation for factual tasks is beyond our scope.

Finally, since multiple questions were derived from a given Wikipedia context in SQuAD, we have the options to create more than 3 related negative answer options for each question. In $MC_{rnd^*}$, we use 6 negative answer options on average ($\pm 2.22$; at most 16) depending on the source context. We find minimal difference from $MC_{rnd}$, highlighting the robustness of 4 overall answer options that is frequently used for MC tasks.

## 4.2 Runtimes

The main motivation for reformulating NLG tasks is to reduce compute time requirements. In Table 5 we present task runtimes in minutes for three Pythia[9] model sizes: 1B, 2.8B and 6.9B. Overall, the MC and LL reformulations bring significant runtime reductions on our benchmarks (2x–176x averaged across model sizes) compared to their NLG counterparts, becoming more and more efficient as model size grows. The most significant improvement was achieved for code generation, where the runtime was reduced from nearly 2 hours[10] to under a minute. As expected LL is more efficient than MC since it has only one answer option to score. These improvements significantly reduce runtime costs during model evaluation, while causing no additional effort when reformulating NLG tasks to LL and only minimal costs when generating random or smart distractors for the MC tasks.

## 4.3 Predicting NLG performance

As mentioned before, we do not aim to completely eliminate NLG tasks but to reduce compute needs during model training by relying on the NLU reformulations. Running occasional NLG evaluations is still beneficial as it gives exact model performance on the NLG formulations. In order to further reduce the frequency of such occasional NLG evaluations, in this section we aim at predicting the NLG performance of a model by training linear regression models. More precisely, to predict the NLG performance of a given model at timestep $i$, we leverage both NLG and NLU performance scores at timesteps $i-1$, $i-2$ and $i-3$ as training data and NLU performance at timestep $i$ as input for the prediction. Table 6 presents absolute prediction error (Err.), and similarly as before, Spearman correlation coefficients of model rank correlations based on the gold and predicted NLG performances. Additionally, we visualize the predicted performance curves in Figure 2.

| | | 1B | 3B | 7B | Avg.(Imp.) |
|---|---|---|---|---|---|
| GSM8K | NLG | 4.77 | 12.55 | 27.08 | 14.80 |
| | MC | 2.45 | 5.88 | 12.90 | 7.08(2.1x) |
| | LL | 1.05 | 2.08 | 3.93 | 2.36(6.3x) |
| Trivia | NLG | 11.23 | 32.40 | 47.90 | 30.51 |
| | MC | 1.13 | 1.93 | 3.75 | 2.27(13.4x) |
| | LL | 1.22 | 1.42 | 1.85 | 1.49(20.5x) |
| SQuAD | NLG | 17.70 | 53.08 | 144.27 | 71.68 |
| | MC | 4.68 | 10.57 | 23.50 | 12.92(5.5) |
| | LL | 1.77 | 3.53 | 7.03 | 4.11(17.4x) |
| Human Eval | NLG | 66.75 | 138.89 | 139.11 | 114.92 |
| | MC | 0.54 | 0.83 | 1.26 | 0.88(130.6x) |
| | LL | 0.48 | 0.66 | 0.83 | 0.65(176.3x) |

Table 5: Task runtimes of a single model checkpoint in minutes on a single Nvidia RTX 6000. For consistency, we present Pythia models only at three different model sizes (1B, 3B and 7B). We present averaged runtime over the three model sizes in columns *Avg.* as well as speed improvements compared to the generative formulation in parenthesis.

| NLG | NLU | Err. | Spearman |
|---|---|---|---|
| GSM8K | MC | 0.031 | 0.48 |
| | $MC_{rnd}$ | 0.038 | 0.43 |
| | LL | 0.021 | 0.62 |
| Trivia | MC | 0.054 | 0.98 |
| | $MC_{rnd}$ | 0.047 | 1.00 |
| | LL | 0.057 | 0.92 |
| SQuAD | $MC_{rnd}$ | 0.046 | 0.93 |
| | LL | 0.064 | 0.95 |
| Human Eval | MC | 0.025 | 0.87 |
| | $MC_{rnd}$ | 0.030 | 0.79 |
| | LL | 0.021 | 0.92 |

Table 6: Results of predicting the NLG performance based on NLU. Err. represents the absolute error between the true and predicted scores, while Spearman indicates model ranking similarity.

As can be seen on Table 6, all three reformulation versions perform on par with each other, achieving 6.4% prediction error at most. As expected, MC and $MC_{rnd}$ perform on par, the former being slightly better, highlighting the advantage of smart distractors. When looking at the average rank correlations of our 8 models we found strong Spearman correlation values on all tasks, except GSM8K which still show moderate correlation. We hypoth-

[9]We only consider Pythia models here, as they feature the same architecture across a wide range of sizes.

[10]Due to the length of the outputs. We use the suggested maximum generation length of 2048.

esize that this is due to the difficult nature, and the low results (under 0.2% on the majority of the training course; Figure 2), of the task, thus model ranking is more prone to noise than in case of other tasks. Overall, predicting NLG performance using only 3 training datapoints proves efficient in following generative model performance more closely, while requiring only few expensive direct NLG benchmark evaluation, striking a balance between precise NLG performance monitoring and compute efficiency.

## 5 Conclusion

We performed an empirical study on the relation between NLG and NLU benchmarks, as well as the possibility to automatically reformulate NLG benchmarks to NLU. Calculating the correlation, we demonstrated that all four benchmarks we used herein had a high correlation between the NLG and NLU variants. Interestingly, the correlation between these variants existed in all MC versions that we tested—both random and smart distractors, and related off-the-shelf NLU benchmarks—and was similarly high. Although LL is slightly less efficient than the MC variants, it is still a valid option that does not need distractor option generation (even though the efforts needed are minimal). Furthermore, we were able to show that runtime could be reduced significantly (2x–176x averaged across model sizes). Hence, we conclude that NLU can be used to estimate NLG model performance to save compute, although we advice against neglecting NLG benchmark evaluation altogether. Furthermore, we also tested whether using only a few NLG evaluations together with NLU formulations is beneficial in NLG performance prediction. We found that using only 3 training datapoints, predicting NLG performance proves efficient for closely tracking generative model quality, reducing the need for frequent costly benchmark evaluations.

## 6 Limitations

We have shown that generative capabilities of small and medium size models can be accessed using reformulation to NLU in various domains. However, it is possible that bigger models (above 7B parameters) behave slightly differently, although we expect these results to hold for them as well. Furthermore, we did not include safety-related domains e.g., ethical evaluation benchmarks, as we consider the models used herein too small to meaningfully handle such complex tasks—capabilities more likely present in larger and/or instruction tuned models. In such cases, the potential for task reformulation would also need to be explored.

## Acknowledgments

This work has been funded by the Free State of Bavaria in the DSgenAI project (Grant Nr.: RMF-SG20-3410-2-18-4). The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the NHR project *ELMOD: Efficient language models for on-device deployment* (Grant Nr.: b239dc). NHR funding is provided by federal and Bavarian state authorities. NHR@FAU hardware is partially funded by the German Research Foundation (DFG) – 440719683. We would like to thank our anonymous reviewers and colleagues for the useful feedback, including: Christian Kroos, Joel Schlotthauer, Lucas Druart, Luzian Hahn and Rishiraj Saha Roy.

## References

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. 2022. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, and 11 others. 2024. Lessons from the trenches on reproducible evaluation of language models. *Preprint*, arXiv:2405.14782.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others.

2021. Evaluating large language models trained on code.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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, and 5 others. 2024. A framework for few-shot language model evaluation.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. OLMo: Accelerating the Science of Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15789–15809.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2024. OLMES: A standard for language model evaluations. *arXiv preprint arXiv:2406.08446*.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. 2025. OLMES: A Standard for Language Model Evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, and 1 others. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing Format Boundaries with a Single QA System. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907.

Toshiki Kuramoto and Jun Suzuki. 2025. Predicting fine-tuned performance on larger datasets before creating them. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 204–212.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. 2023. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, and 1 others. 2024. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D Manning, Christopher Re, Diana Acosta-Navas, Drew A. Hudson, and 31 others. 2023. Holistic Evaluation of Language Models. *Transactions on Machine Learning Research*.

Zhengzhong Liu, Aurick Qiao, Willie Neiswanger, Hongyi Wang, Bowen Tan, Tianhua Tao, Junbo Li, Yuqi Wang, Suqi Sun, Omkar Pangarkar, Richard Fan, Yi Gu, Victor Miller, Yonghao Zhuang, Guowei He, Haonan Li, Fajri Koto, Liping Tang, Nikhil Ranjan, and 8 others. 2024. LLM360: Towards Fully Transparent Open-Source LLMs. In *First Conference on Language Modeling*.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D. Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, Noah A. Smith, Pang Wei Koh, and Jesse Dodge. 2025. Datadecide: How to predict best pretraining data with small experiments. *Preprint*, arXiv:2504.11393.

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. 2023. OctoPack: Instruction Tuning Code Large Language Models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Aidar Myrzakhan, Sondos Mahmoud Bsharat, and Zhiqiang Shen. 2024. Open-llm-leaderboard: From multi-choice to open-style questions for llms evaluation, benchmark, and arena. *arXiv preprint arXiv:2406.07545*.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, and 1 others. 2024. 2 OLMo 2 Furious. *arXiv preprint arXiv:2501.00656*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. 2024. Efficient benchmarking (of language models). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2519–2536.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. 2024. tinybenchmarks: evaluating LLMs with fewer examples. In *Forty-first International Conference on Machine Learning*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don‘t know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Rylan Schaeffer, Hailey Schoelkopf, Brando Miranda, Gabriel Mukobi, Varun Madan, Adam Ibrahim, Herbie Bradley, Stella Biderman, and Sanmi Koyejo. 2024. Why Has Predicting Downstream Capabilities of Frontier AI Models with Scale Remained Elusive? In *ICML 2024 Next Generation of AI Safety Workshop*.

Xiaoqiang Wang, Lingfei Wu, Tengfei Ma, and Bang Liu. 2024a. FAC$^2$E: Better understanding large language model capabilities by dissociating language and cognition. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13228–13243.

Xunzhi Wang, Zhuowei Zhang, Qiongyu Li, Gaonan Chen, Mengting Hu, Bitong Luo, Hang Gao, Zhixin Han, Haotian Wang, and 1 others. 2024b. Ubench: Benchmarking uncertainty in large language models with multiple choice questions. *arXiv preprint arXiv:2406.12784*.

Chaojun Xiao, Jie Cai, Weilin Zhao, Guoyang Zeng, Biyuan Lin, Jie Zhou, Zhi Zheng, Xu Han, Zhiyuan Liu, and Maosong Sun. 2024. Densing law of llms. *Preprint*, arXiv:2412.04315.

Ziyin Zhang, Zhaokun Jiang, Lizhen Xu, Hongkun Hao, and Rui Wang. 2024. Multiple-choice questions are efficient and robust llm evaluators. *arXiv preprint arXiv:2405.11966*.

Kaitlyn Zhou, Su Lin Blodgett, Adam Trischler, Hal Daumé III, Kaheer Suleman, and Alexandra Olteanu. 2022. Deconstructing NLG evaluation: Evaluation practices, assumptions, and their implications. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–324, Seattle, United States. Association for Computational Linguistics.

# A Prompt for Smart Distractor Creation

**Prompt:**
Please use the given question `{question}` and create 4 answers, the first one being `{correct_answer}`, the correct answer, and the other three being incorrect answers. Use `JSONL` to respond.

**Examples for TriviaQA:**

- `question="Which American-born Sinclair won the Nobel Prize for Literature in 1930?", answers=["Sinclair Lewis", "Upton Sinclair", "Sinclair Ferguson", "Sinclair Smith"]`
- `question="Where in England was Dame Judi Dench born?", answers=["York", "London", "Manchester", "Oxford"]`
- `question="When did the founder of Jehovah's Witnesses say the world would end?", answers=["1914", "2012", "1844", "1975"]`
- `question="1998 was the Chinese year of which creature?", answers=["tiger", "rabbit", "dragon", "giraffe"]`
- `question="The first credit cards were for use in what type of establishments?", answers=["restaurants", "cinemas", "gas stations", "hotels"]`

**Examples for GSM8K:**

- `question="Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?", answers=[ "Natalia sold 48/2 = «48/2=24»24 clips in May. Natalia sold 48+24 = «48+24=72»72 clips altogether in April and May. #### 72", "Natalia sold 48/2 = «48/2=24»24 clips in`

May. Natalia sold 48 + 20 = «48+20=68»68 clips altogether in April and May. #### 68",
"Natalia sold 48/2 = «48/2=24»24 clips in May. Natalia sold 48 + 22 = «48+22=70»70 clips altogether in April and May. #### 70",
"Natalia sold 48 × 2 = «48*2=96»96 clips in May. Natalia sold 48 + 96 = «48+96=144»144 clips altogether in April and May. #### 96"]

- question="Weng earns $12 an hour for babysitting. Yesterday, she just did 50 minutes of babysitting. How much did she earn?",
  answers=[
  "Weng earns 12/60 = «12/60=0.2 »0.2 per minute. Working 50 minutes, she earned 0.2 x 50 = «0.2*50=10»10. #### 10",
  "Weng earns 12/60 = «12/60=0.2 »0.2 per minute. Working 50 minutes, she earned 0.2 × 60 = «0.2*60=12 »12. #### 12",
  "Weng earns 12/60 = «12/60=0.2 »0.2 per minute. Working 50 minutes, she earned 0.2 × 40 = «0.2*40=8»8. #### 8,
  "Weng earns 12/60 = «12/60=0.2 »0.2 per minute. Working 50 minutes, she earned 0.2 × 45 = «0.2*45=9 »9. #### 9"]

- question="Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?",
  answers=[
  "In the beginning, Betty has only 100 / 2 = $«100/2=50 »50. Betty's grandparents gave her 15 * 2 = $«15*2=30 »30. This means, Betty needs 100 - 50 - 30 - 15 = $«100-50-30-15=5 »5 more. #### 5",
  "In the beginning, Betty has only 100 / 2 = $«100/2=50 »50. Betty's grandparents gave her 15 * 2 = $«15*2=30 »30. This means, Betty needs 100 - 50 - 30 = $«100-50-30=20 »20 more. #### 20",
  "In the beginning, Betty has only 100 / 2 = $«100/2=50 »50. Betty's grandparents gave her 15 * 2 = $«15*2=30 »30. This means, Betty needs 100 - 50 - 15 = $«100-50-15=35 »35 more. #### 35",
  "In the beginning, Betty has only 100 / 2 = $«100/2=50 »50. Betty's grandparents gave her 15 * 2 = $«15*2=30 »30. This means, Betty needs 100 - 30 - 15 = $«100-30-15=55 »55 more. #### 55"]

- question="James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?",
  answers=[
  "He writes each friend 3*2= «3*2=6 »6 pages a week. So he writes 6*2= «6*2=12 »12 pages every week. That means he writes 12*52= «12*52=624 »624 pages a year. #### 624",
  "He writes each friend 3×2 = «32=6 »6 pages a week.. So he writes 6×2 = «62=12 »12 pages every week. That means he writes 12×50 = «12*50=600 »600 pages a year. #### 600",
  "He writes each friend 3×2 = «32=6 »6 pages a week. So he writes 6×1 = «61=6 »6 pages every week. That means he writes 6×52 = «6*52=312 »312 pages a year. #### 312",
  "He writes each friend 3×2 = «32=6 »6 pages a week. So he writes 6×2 = «62=12 »12 pages every week. That means he writes 12×12 = «12*12=144 »144 pages a year. #### 144"]

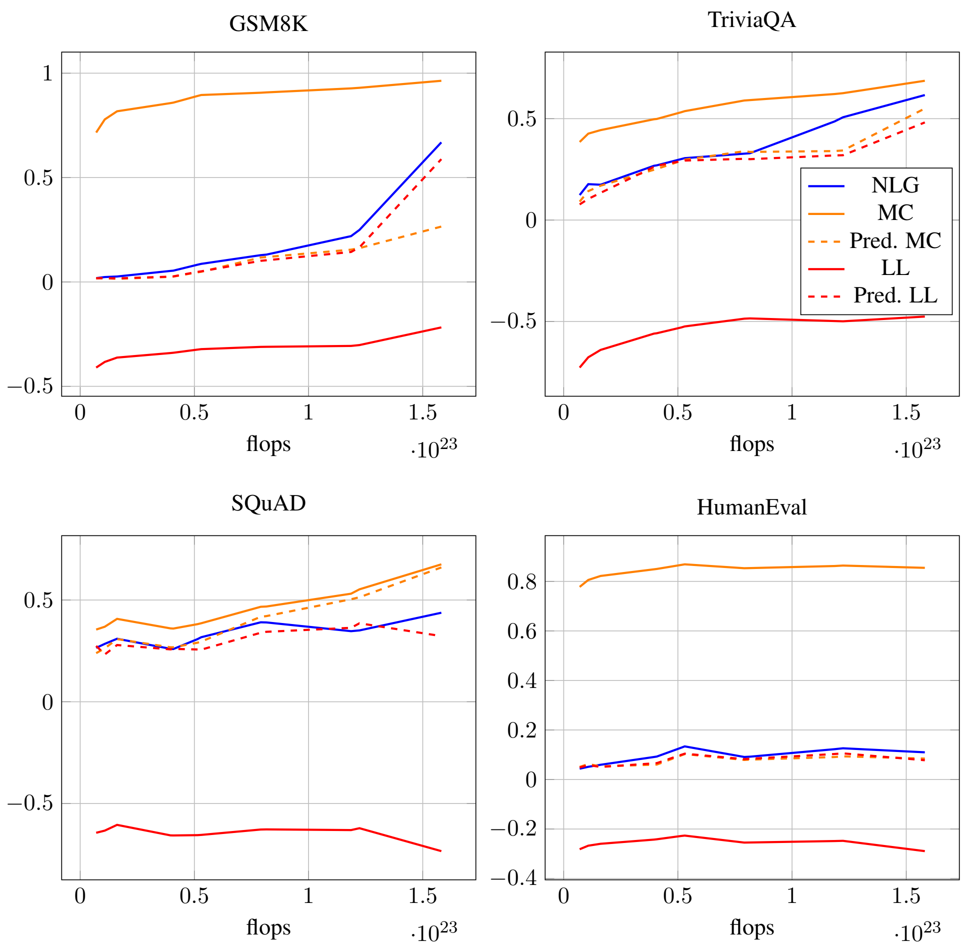


Figure 2: Predicted (Pred.) generative performance using MC or LL performance. Each predicted point is based on 3 previous NLU and NLG reference points using linear regression.

| NLG | NLU | $P_{macro}$ | $P_{micro}$ | Spearman |
|---|---|---|---|---|
| GSM8K | MC | 0.75(0.12) | 0.52(0.00) | 0.76(0.03) |
| | $MC_{rnd}$ | 0.76(0.11) | 0.57(0.00) | 0.76(0.03) |
| | LL | 0.79(0.09) | 0.56(0.00) | 0.81(0.01) |
| | $MC_{ao}$ | 0.38(0.26) | 0.90(0.00) | 0.88(0.00) |
| | $LL_{ao}$ | 0.22(0.38) | -0.04(0.80) | -0.07(0.87) |
| Trivia | MC | 0.90(0.03) | 0.94(0.00) | 0.86(0.01) |
| | $MC_{rnd}$ | 0.91(0.02) | 0.88(0.00) | 0.98(0.00) |
| | LL | 0.90(0.03) | 0.69(0.00) | 0.81(0.01) |
| | MMLU | 0.89(0.04) | 0.94(0.00) | 0.95(0.00) |
| SQuAD | $MC_{rnd}$ | 0.90(0.03) | 0.88(0.00) | 0.93(0.00) |
| | $MC_{rnd^*}$ | 0.90(0.03) | 0.84(0.00) | 0.93(0.00) |
| | LL | 0.65(0.15) | 0.85(0.00) | 0.69(0.06) |
| | BoolQ | 0.78(0.11) | 0.24(0.09) | 0.45(0.26) |
| Human Eval | MC | 0.83(0.09) | 0.79(0.00) | 0.81(0.02) |
| | $MC_{rnd}$ | 0.85(0.07) | 0.75(0.00) | 0.81(0.02) |
| | LL | 0.86(0.07) | 0.73(0.00) | 0.79(0.03) |
| cpp | MC | 0.87(0.06) | 0.86(0.00) | 0.74(0.04) |
| | $MC_{rnd}$ | 0.92(0.03) | 0.83(0.00) | 0.90(0.00) |
| | LL | 0.92(0.03) | 0.77(0.00) | 0.81(0.01) |
| go | MC | 0.67(0.22) | 0.71(0.00) | 0.76(0.03) |
| | $MC_{rnd}$ | 0.68(0.21) | 0.74(0.00) | 0.81(0.01) |
| | LL | 0.66(0.24) | 0.72(0.00) | 0.88(0.00) |
| java | MC | 0.85(0.10) | 0.86(0.00) | 0.98(0.00) |
| | $MC_{rnd}$ | 0.86(0.08) | 0.80(0.00) | 0.83(0.01) |
| | LL | 0.90(0.05) | 0.64(0.00) | 0.67(0.07) |
| js | MC | 0.92(0.02) | 0.80(0.00) | 0.83(0.01) |
| | $MC_{rnd}$ | 0.86(0.06) | 0.79(0.00) | 0.88(0.00) |
| | LL | 0.90(0.04) | 0.75(0.00) | 0.71(0.05) |
| python | MC | 0.92(0.02) | 0.81(0.00) | 0.74(0.04) |
| | $MC_{rnd}$ | 0.91(0.02) | 0.81(0.00) | 0.79(0.02) |
| | LL | 0.93(0.01) | 0.81(0.00) | 0.81(0.01) |
| rust | MC | 0.77(0.14) | 0.67(0.00) | 0.79(0.02) |
| | $MC_{rnd}$ | 0.87(0.04) | 0.52(0.00) | 0.62(0.10) |
| | LL | 0.87(0.05) | 0.70(0.00) | 0.83(0.01) |

Table 7: Correlation statistics of the NLG tasks and their various reformulated formats. We present p-values in parentheses. The NLG task *Code* represents HumanEval results averaged over the 6 coding languages (cpp, go, java, js, python and rust). MC stands for the multiple-choice formulation with smart distractors, $MC_{rnd}$ uses random distractors, while LL is the log-likelihood formulation. For GSM8K we tested setups where only the final answer has to be scored by the model ($*_{ao}$) as proposed by (Zhang et al., 2024). In case of TriviaQA and SQuAD, we tested cross-benchmark pairings: MMLU and BoolQ respectively. Additionally, in $MC_{rnd^*}$ we used more than 4 answer options for SQuAD.

| model | #params. | #tokens | checkpoints |
|---|---|---|---|
| EleutherAI/pythia-1b-deduped | 1B | 300B | step10000, step20000, step30000, step50000, step143000 |
| EleutherAI/pythia-2.8b-deduped | 2.8B | 300B | step10000, step20000, step30000, step50000, step143000 |
| EleutherAI/pythia-6.9b-deduped | 6.9B | 300B | step10000, step20000, step30000, step50000, step143000 |
| allenai/OLMo-1B-0724-hf | 1B | 3.05T | step10000-tokens20B, step20000-tokens41B, step29000-tokens60B, step50000-tokens104B, step150000-tokens314B, step500000-tokens1048B, step621000-tokens1301B, step1000000-tokens2096B, step1454000-tokens3048B |
| allenai/OLMo-7B-0724-hf | 7B | 2.75T | step10000-tokens41B, step14500-tokens60B, step25500-tokens106B, step75000-tokens314B, step250000-tokens1048B, step310000-tokens1300B, step500000-tokens2097B, step650650-tokens2729B |
| LLM360/Amber | 7B | 1.25T | ckpt_005, ckpt_010, ckpt_016, ckpt_027, ckpt_082, ckpt_275, ckpt_358 |
| LLM360/Crystal | 7B | 1.4T | CrystalCoder_phase1_checkpoint_006000, CrystalCoder_phase1_checkpoint_009000, CrystalCoder_phase1_checkpoint_012000, CrystalCoder_phase1_checkpoint_021000, CrystalCoder_phase1_checkpoint_067500, CrystalCoder_phase2_checkpoint_015000, CrystalCoder_phase3_checkpoint_027728 |
| allenai/OLMo-2-1124-7B | 7B | 4T | stage1-step10000-tokens42B, stage1-step14000-tokens59B, stage1-step25000-tokens105B, stage1-step75000-tokens315B, stage1-step250000-tokens1049B, stage1-step310000-tokens1301B, stage1-step500000-tokens2098B, stage1-step720000-tokens3020B, stage2-ingredient1-step11931-tokens50B |

Table 8: Details of the used models. The model and checkpoint names are references to the content on the Huggingface Hub, while the number of parameters and training tokens are based on the respective model publications.

| | benchm. | (correct) answer | distractors | specifics |
|---|---|---|---|---|
| GSM8K | question | Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May? | | |
| | original | Natalia sold 48/2 = <<48/2=24>>24 clips in May. Natalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May. #### 72 | | |
| | MC | | [Natalia sold 48 × 2 = <<48*2=96>>96 clips in May. Natalia sold 48 + 96 = <<48+96=144>>144 clips altogether in April and May. #### 144, Natalia sold [...]] | |
| | MC$_{rnd}$ | | [Weng earns 12/60 = «12/60=0.2»..., He writes each friend [. . . ]] | |
| | GSM-MC | 72 | [64, 61, 89] | |
| Trivia | question | Which was the first European country to abolish capital punishment? | | |
| | original | Norway | | aliases: Norvège, Mainland Norway, Norwegian state, [. . . ] normalized_aliases: norwegen, kongeriket norge, norway, [. . . ] |
| | MC | | [Germany, Italy, Poland] | |
| | MC$_{rnd}$ | | [Chicago Bears, Ballet, 6] | |
| SQuAD | question | Who is the main character in "Childe Harold's: Canto I?" | | |
| | original | no answer in this context | | context: [. . . ] Studying and analyzing literature becomes very important in terms of learning about our history. [. . . ] Lord Byron talks about the Spanish and the French in "Childe Harold's Pilgrimage: Canto I" [. . . ] |
| | question | We can learn what by carefully examining our literature? | | |

Continued on next page

Table 9 – continued from previous page

| | benchm. | answer | distractors | specifics |
|---|---|---|---|---|
| | original | our history | | context: [. . . ] Studying and analyzing literature becomes very important in terms of learning about our history. [. . . ] Lord Byron talks about the Spanish and the French in “Childe Harold’s Pilgrimage: Canto I” [. . . ] |
| | $MC_{rnd}$ | | [Lord Byron, written records, corpse] | answers taken from same context |
| HumanEval | question | [...] Given a string, find out how many distinct characters (regardless of case) does it consist of [...] | | |
| | original | return len(set(string.lower())) | | buggy solution: return set(len(string.lower())) |
| | MC | | return set(len(string.lower())) | |
| | $MC_{rnd}$ | | [return ’ ’.join([str(x) for x in range(n + 1)]), [...]] | |
| | MMLU | | | |
| | question | Find the degree for the given field extension Q(sqrt(2), sqrt(3), sqrt(18)) over Q. | | |
| | | 4 | [0,2,6] | |
| | BoolQ | | | |
| | question | do iran and afghanistan speak the same language | | |
| | | TRUE | FALSE | |

Table 9: Example items from the used benchmarks and our reformulations. GSM-MC references (Zhang et al., 2024). As MMLU and BoolQ were used as MC pairings of other benchmarks, they only have the original version.