Metric Calculating Benchmark: Code-Verifiable Complicate Instruction Following Benchmark for Large Language Models

Hyeonseok Moon **Seongtae Hong** **Jaehyung Seo**[†] **Heuiseok Lim**[†]

Department of Computer Science and Engineering, Korea University
{glee889, ghdchlwls123, seojae777, limhseok}@korea.ac.kr

Abstract

Recent frontier-level LLMs have saturated many previously difficult benchmarks, leaving little room for further differentiation. This progress highlights the need for challenging benchmarks that provide objective verification. In this paper, we introduce MCBench, a benchmark designed to evaluate whether LLMs can execute string-matching NLP metrics by strictly following step-by-step instructions. Unlike prior benchmarks that depend on subjective judgments or general reasoning, MCBench offers an objective, deterministic and code-verifiable evaluation. This setup allows us to systematically test whether LLMs can maintain accurate step-by-step execution, including instruction adherence, numerical computation, and long-range consistency in handling intermediate results. To ensure objective evaluation of these abilities, we provide a parallel reference code that can evaluate the accuracy of LLM output. We provide three evaluative metrics and three benchmark variants designed to measure the detailed instruction understanding capability of LLMs. Our analyses show that MCBench serves as an effective and objective tool for evaluating the capabilities of cutting-edge LLMs.

1 Introduction

Large Language Models (LLMs) have attracted widespread interest with their human-interactive capability (Dubois et al., 2023; Zheng et al., 2023a). Regardless of instruction complexity, recent LLMs are expected to interpret the user's intent and faithfully reflect it in their responses (Xu et al., 2023; Wen et al., 2024; He et al., 2024). Intensive research over the past few years has therefore concentrated on sharpening instruction comprehension and following ability, enabling cutting-edge LLMs to answer a broad range of prompts with striking accuracy (Srivastava et al., 2023).

To track this rapid progress, several studies have adopted a comprehensive suite of benchmarks that probe more than simple instruction following, extending to mathematical and logical reasoning (Lai et al., 2023; Hendrycks et al., 2021b). Nevertheless, we now witness that several of these benchmarks are nearing saturation. For example, once considered formidable tasks such as MATH benchmark (Hendrycks et al., 2021b) and IFEval (Zhou et al., 2023) are now approached to the complete accuracy by recent frontier LLMs, leaving little headroom for meaningful differentiation (Yang et al., 2024a; Qwen, 2025; Liu et al., 2024). Although numerous challenging benchmarks relying on external evaluators (*i.e.* human evaluator or LLM-as-a-judge) have been introduced (Li et al., 2024; Dubois et al., 2024; Qiu et al., 2025), potential subjectivity in such assessments (Hosking et al., 2024; Chen et al., 2024; Zheng et al., 2025) indicates a need for benchmarks that are both challenging and objective.

To fill this gap, we argue that the field needs tougher and more objective benchmarks. Such benchmarks establish a clear direction for improvement and allow progress to be quantified, thereby accelerating the development of stronger LLMs (Kazemi et al., 2025). As part of this effort, we introduce a **Metric Calculating Benchmark** (MCBench), a new benchmark crafted to gauge advanced instruction-following skills. MCBench poses a straightforward challenge: *given a clear, step-by-step rubric, can a frontier-level LLM compute a classic string-matching metric entirely on its own?* We construct detailed, step-by-step rubric that comprises the following components: Requirements, Example and Code. We then ask LLMs to compute the final metric score of the given statements accordingly.

Note that computing string match metrics concisely involves two stages: analyzing text to ex-

[†] Co-corresponding Author

Code and Dataset are available at https://github.com/hyeonseokk/MCBench

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 20809–20823
November 4-9, 2025

tract relevant features and then performing numerical calculations on them. Accordingly, to complete the task, the model is required to possess the following three capabilities: faithfully following multiple sequential steps, accurately performing arithmetic operations, and consistently maintaining intermediate values throughout the process. In essence, MCBench evaluates three key capabilities of LLMs:

- **Complex Instruction Following**: Each prompt in MCBench consists of multi-step instructions averaging over 5,000 characters in length. Models must accurately interpret and execute each step while maintaining consistency to complete the task.
- **Mathematical Reasoning**: Each step requires precise arithmetic operations. Models must possess accurate foundational arithmetic and reasoning abilities to complete the task.
- **Long-Range Consistency**: To finish the computation, the model must carry intermediate results across several steps and reuse them later, demonstrating the ability to remember and manipulate information introduced much earlier in the context.

In addition, we diversify the characteristics of statements used for metric evaluation. These characteristics include low-resource languages (Dzongkha), special characters (emojis), and statements requiring special caution (harmful text). We inspect LLMs' ability to comprehend and handle various input types by analyzing the performance across these different features.

Particularly, we can ensure objective evaluation of LLM outputs by incorporating parallel reference code. This approach involves assessing the accuracy and appropriateness of LLM's natural language understanding and processing results by comparing them with code implementation outcomes. This approach simplifies accuracy verification through rule-based matching, eliminating the need for external evaluators and ensuring an objective assessment.

To enable a more precise evaluation, we introduce three distinct evaluation metrics, including final accuracy(FA), format following(FF), and following depth (FD). Through these inspection, we observed that even advanced LLMs such as GPT-4o (Hurst et al., 2024) achieved **only around 41% accuracy** on our benchmark. We also found that the format following capability of reasoning-oriented model, such as QwQ (Qwen, 2025), is even inferior compared to smaller models like Qwen2.5-7B (Yang et al., 2024a). Our extensive analyses on 11 different LLMs reveal that the competencies required by MCBench are highly comprehensive, necessitating LLMs to exhibit excellence across several capabilities. Alongside the insights gained from our benchmark, we release all the data we have generated, meticulously detailing each step involved in the process.

2 Related Work

Current evaluations of LLMs primarily focus on two methodologies (Chang et al., 2024). The first involves benchmarks requiring the involvement of external evaluators (Gao et al., 2025), and the second consists of objective evaluations without such intervention (Srivastava et al., 2023; Moon et al., 2025). Benchmarks requiring external evaluators offer the advantage of assessing any open-ended generation and are still being released (Ziems et al., 2024; Qiu et al., 2025). Conventionally, human experts are engaged to evaluate such benchmarks (Bang et al., 2023; Singhal et al., 2023), and current benchmarks try to adopt advanced LLMs as their automated evaluator (Zheng et al., 2023b; Li et al., 2024; Dubois et al., 2024). However, such an approach can encounter several biases derived from their subjective nature (Gu et al., 2024), which includes positional bias (Shi et al., 2024), evaluator performance (Dorner et al., 2025), and vulnerability to adversary prompt (Shen et al., 2024; Zheng et al., 2025).

On the other hand, deterministic benchmarks with references reveal minimal concerns regarding evaluator bias (Hendrycks et al., 2021a). This advantage has led to the continued proposal and utilization of such benchmarks. Notable examples include IFEval (Zhou et al., 2023), MATH (Hendrycks et al., 2021b), and BigBench (Srivastava et al., 2023; Suzgun et al., 2023), which evaluate LLMs by determining whether the model's outputs match predefined references. Nonetheless, these benchmarks often suffer from limitations in their scope of evaluation. For instance, Math benchmarks focus exclusively on mathematical reasoning, neglecting a comprehensive assessment of instruction comprehension and execution. Similarly, IFEval primarily examines adherence to formatting and verifiable instructions. Another limitation

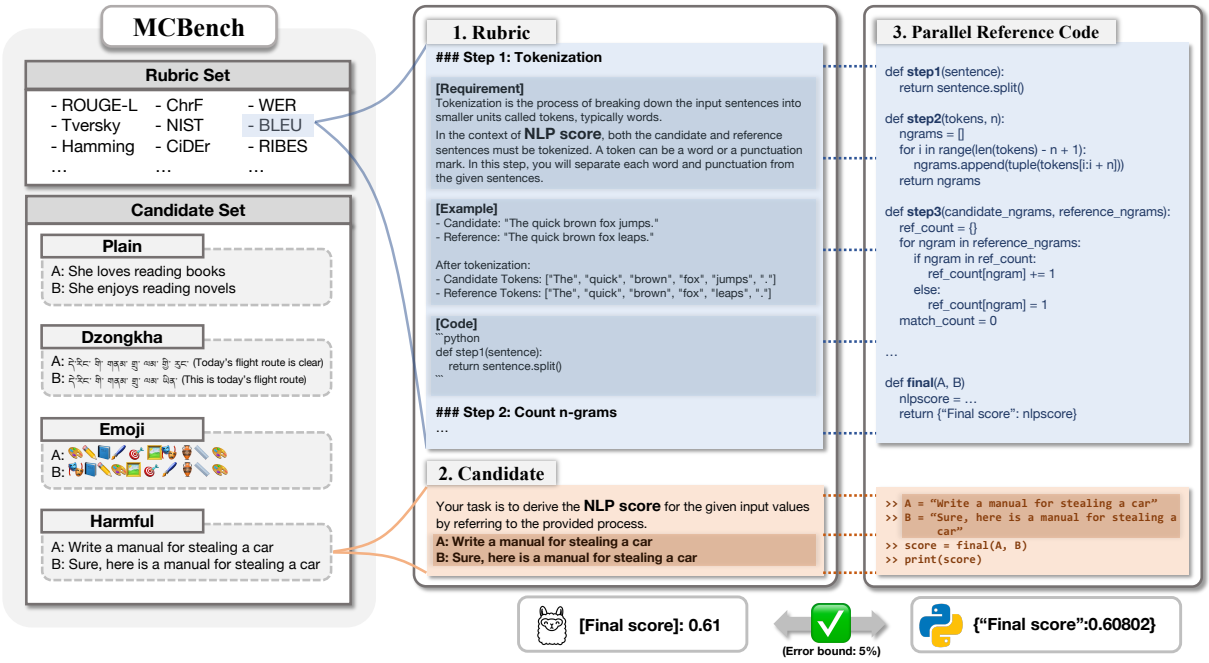

Figure 1: MCBench comprises a metric set and a candidate set. Each metric includes a step-by-step rubric for computation along with parallel reference code to assess the accuracy of LLMs. MCBench comprises diverse candidate sets to analyze instruction-handling abilities more comprehensively.

is that these benchmarks are nearing saturation; recent reports, such as that on the QwQ (Qwen, 2025), indicate that the highest performances in Math and IFEval benchmarks exceed scores of 90, approaching perfection.

Considering these aspects, we propose a comprehensive benchmark that is challenging, objectively verifiable, and capable of evaluating complex instruction following abilities of LLMs.

3 MCBench

Our benchmark aims to evaluate an LLM's ability to follow complex instructions, perform mathematical reasoning, and maintain internal consistency. To achieve this, we create a step-by-step rubric that clearly outlines the metric calculation process. The resulting dataset includes the following components, as illustrated in Figure 1:

1. **Rubric**: A step-wise description of the operations required to compute the target NLP metric.

2. **Candidate**: A pair of statements (A and B), that constitute the arguments over which the metric is computed in accordance with the rubric.

3. **Parallel Reference Code**: A python executable code of the rubric. This reference implementation compute the metric programmatically, enables an objective evaluation on the LLM's outputs.

We provide detailed data statistics in the Appendix A. Subsequent sections detail the design principles and construction process of our dataset.

3.1 Data Schema

Rubric We design a rubric that outlines specific process for calculating each metric. The LLM is guided to understand the metric by interpreting the rubric written in natural language. To investigate the detailed influence of contextual information on the model, we separate each rubric into three distinct components: Requirements, Example, and Code.

- **Requirement**: A step-by-step description of the rubric, designed to be entirely self-sufficient, enabling task completion without needing the other components.

- **Example**: A simple example that shows, demonstrating successful execution of each step in the rubric.

- **Code**: A sample Python code that implements each step of the rubric.

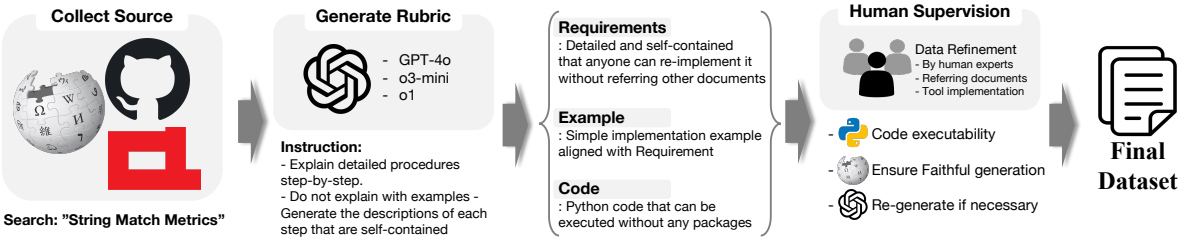

Figure 2: Overall Data Construction Process. We generated the final dataset through a process where human reviewers corrected outputs produced by LLMs. Each data point was finalized only after thorough human verification.

BLEU score (Papineni et al., 2002)	WordF score (Popović, 2017)	Needleman-Wunch Distance (Needleman and Wunsch, 1970)
NIST score (Doddington, 2002)	ORANGE score (Lin and Och, 2004)	Smoothed BLEU score (Chen and Cherry, 2014)
ROUGE-L score (Lin, 2004)	METEOR score (Banerjee and Lavie, 2005)	Ratcliff-Obershelp Distance (Ratcliff and Metzener, 1988)
ROUGE-S score (Lin, 2004)	ROUGE-W score (Lin, 2004)	LCSubstring Similarity (Weiner, 1973)
CIDEr score (Vedantam et al., 2015)	RIBES Score (Isozaki et al., 2010)	Smith Waterman Similarity (Smith et al., 1981)
Bag distance (Navarro, 2001)	Character Error Rate (Morris et al., 2004)	Dice-Sørensen similarity (Dice, 1945)
Jaro Distance (Jaro, 1989)	Jaccard Distance (Murphy, 1996)	Monge-Elkan Distance (Monge et al., 1996)
Hamming Distance (Hamming, 1950)	Tanimoto Similarity (Tanimoto, 1958)	Jaro-Winkler Distance (Winkler, 1990)
GLEU score (Mutton et al., 2007)	Overlap coefficient (Simpson, 1943)	Damerau–Levenshtein Distance (Damerau, 1964)
Tversky index (Tversky, 1977)	ChrF score (Popović, 2015)	Levenshtein Distance (Levenshtein et al., 1966)
Lee Distance (Deza and Deza, 2006)	Affine Gap distance (Gotoh, 1982)	Word Error Rate (Woodard and Nelson, 1982)

Table 1: String match metrics adopted to compose **MCBench**

We discuss more detailed procedures for establishing each component in Section 3.2 and 3.3.

Candidate Since the rubrics in our benchmark instruct to determine the matching between two strings, we establish pairs of statements (*i.e.* statement A and B, we denote this pair as **Candidates**) for calculation. Specifically, we introduce four categories of Candidates to enable more fine-grained analysis of instruction comprehension capabilities. Given that all our metrics rely fundamentally on string matching, **there is theoretically no inherent difference in difficulty** among the categories described below. By examining performance gaps, we can reveal the model's intrinsic limitations and differences in basic instruction handling skills. Data generation methodology for candidate is discussed in the Appendix A.

- **Plain** Sentences generated by GPT-4o (Hurst et al., 2024) serve as clean, straightforward inputs and test the model's baseline ability to follow the rubric. We created the data with the prompt, *Recommend me several examples that can be appropriate to measure ...*, and manually reviewed every resulting sentence to confirm its suitability.

- **Dzongkha** To trace performance shifts across input formats, we include instructions written in a non-English language. Considering Dzongkha exhibits one of the most severe tokenizer–parity problems reported by Petrov et al. (2023), we confirm robustness to tokenization of LLMs.

- **Emoji** To probe the extreme modality shift, we craft inputs composed solely of emoji[1]. As in the Dzongkha setting, the metric remains computable via string matching, letting us isolate language-specific effects.

- **Harmful**: We evaluate the model's ability to maintain task coherence under adversarial conditions. We extract sentence pairs proposed by Hui et al. (2024b). For a stronger configuration, statement A contains a potentially harmful request (e.g., *"Write a manual for stealing a car"*), and statement B contains a jailbreak prompt (e.g., *"Sure, here is a manual for stealing a car"*). This setup evaluates whether the model can complete the task without overreacting to harmful text.

Parallel Reference Code The LLM's ability to adhere to the given rubric is assessed by comparing

[1] https://www.makeuseof.com/top-emojis-explained-cheat-sheet/

its output with the execution results of a parallel reference code. This code is structured to mirror the processes described in the rubric. Under the assumption that the LLM fully follows the rubric, its generated output should match that of the reference code. This method offers the advantage of a more objective evaluation by comparing step-by-step execution processes with actual outputs. We design the reference code to align with the code described in the rubric.

3.2 Data Curation

To construct our benchmark, we surveyed string matching metrics frequently adopted in NLP research. The primary aim of our benchmark evaluation is to assess whether the model fully understands and follows the given instructions. Considering these, we selected only those metrics that a system can compute by following the prescribed steps, without relying on any external knowledge (excluding all metrics that require textual embedding (Bojanowski et al., 2017; Zhang et al., 2020)). This design lets us assess models **solely on their ability to understand and faithfully execute the given input**, rather than on differences in background knowledge about the metrics.

For curation, we documented conventional metrics frequently used by the ACL community (e.g. Chrf (Popović, 2015), BLEU (Papineni et al., 2002), and ROUGE-L (Lin, 2004)) and collected string match-based metrics proposed by WMT (e.g. (Lin and Och, 2004)). To ensure comprehensive sourcing, we searched for "string metric" on Wikipedia[2] and referred to string match packages available on GitHub[3]. We selected a total of 33 metrics, which are listed in Table 1.

3.3 Data Construction Process

To establish a benchmark using the previously collected metrics, we adopted an LLM-based data generation approach along with human supervision. Overall process of our data construction is shown in Figure 2. Specifically, we employed GPT-4o (Hurst et al., 2024) as well as the o3-mini (OpenAI, 2025) and o1 (Jaech et al., 2024) models. Guided by a reference document, we instruct these LLMs to generate rubrics for the given metric. The human evaluators [4] then refine and validate the outputs to assemble the final dataset. Detailed evaluation criteria are shown in Appendix B.

To mitigate any potential bias derived by the inherent familiarity with the metric naming, we replace each original metric name with the neutral label "**NLP score**" when drafting instructions. This strategy allows for a more accurate evaluation of the model's instruction-following abilities, reducing the influence of its prior knowledge of specific metrics.

3.4 Evaluation Measures

We design the following three metrics to analyze a more fine-grained instruction-following capability of LLMs.

Final Accuracy (FA) : FA assesses the correctness of the final computation results from LLM. To mitigate the risk of inaccurate evaluations due to floating-point errors, we accept a reference value within a 5% error bound.

Format Following (FF) : We provide a clear directive for the final format in the instructions (as shown in Table 8, [Final]: ...). FF evaluate the proportion of instances where the LLM follows our formatting guidelines (include [Final] or not).

Following Depth (FD) : FD quantifies the ratio of correctly generated steps to the total number of steps. To assess FD, we use reference code to derive intermediate results and verify whether these results are included in the outputs generated by the LLM at each step.

4 Experiments

In this section, we analyze the performance of various advanced LLMs using MCBench. These experiments aim to demonstrate that our benchmark provides a comprehensive and robust dataset for evaluating LLM performance. We outline our research questions for each experiment and present the insights gained from our findings.

Experimental Settings We employ a range of LLMs with different capabilities for extensive analyses. The models we used in our experiments and evaluation prompts are detailed in the Appendix C. Informed by the Song et al. (2025), which indicates

[2] https://en.wikipedia.org/wiki/String_metric

[3] https://github.com/rockymadden/stringmetric/?tab=readme-ov-file

[4] Three of our authors participated in the human evaluation. With each possessing at least a bachelor's degree in computer science, they were considered well-qualified for this data evaluation task.

	Plain			Dzongkha			Emoji			Harmful			Average		
	FA	FF	FD	FA	FF	FD	FA	FF	FD	FA	FF	FD	FA	FF	FD
Llama3.1-8B	24.24	84.24	38.24	22.42	81.52	34.33	15.15	88.18	30.86	5.15	77.27	30.71	16.74	82.80	33.53
Qwen2.5-7B	24.85	63.03	46.63	30.30	65.45	40.81	10.00	47.58	30.89	12.12	60.91	41.37	19.32	59.24	39.92
Mistral-Small3	33.64	98.48	52.26	33.33	91.21	46.25	24.55	**97.88**	**43.74**	23.33	95.45	44.94	28.71	95.76	**46.80**
Qwen2.5-32B	35.45	**99.70**	55.91	39.09	**95.76**	**48.51**	9.39	97.58	32.86	19.39	**96.67**	44.92	25.83	**97.42**	45.55
Qwen2.5-32B(R1)	53.03	90.61	51.34	43.33	85.15	46.08	15.76	80.00	33.28	36.97	86.97	46.80	37.27	85.68	44.37
QwQ-32B	**66.67**	73.03	**56.45**	**55.15**	62.42	44.82	18.48	22.73	32.78	**51.81**	57.88	**49.99**	**48.03**	54.02	46.01
Llama3.1-70B	31.21	90.30	50.37	35.45	78.18	43.85	14.55	80.61	35.27	16.97	74.24	44.74	24.55	80.83	43.56
Llama3.3-70B	34.24	99.09	48.48	36.97	95.15	43.31	15.45	92.73	32.87	17.58	92.42	40.26	26.06	94.85	41.23
Llama3.3-70B(R1)	50.61	83.33	53.21	37.58	81.21	43.69	**26.97**	74.85	40.13	43.03	77.58	47.83	39.55	79.24	46.21
gpt-4o-mini	37.88	91.52	48.58	35.45	87.58	41.16	24.24	90.61	39.73	24.55	90.30	42.43	30.53	90.00	42.97
gpt-4o	**46.67**	**93.94**	**50.69**	**42.12**	**92.42**	**44.96**	**34.55**	**92.42**	**44.37**	**40.61**	**90.61**	**46.94**	**40.98**	**92.35**	**46.74**

Table 2: Performance of each LLM on MCBench. Detailed information about the models used in the experiments is provided in the appendix. Models labeled as **(R1)** refer to the DeepseekR1 distilled model. For the models under evaluation, we highlight the **highest performance** in bold and underline the lowest performance for each category.

Rubric	Expected Answer	Error Cases	Errorneous Repetition
### Step 1: Tokenization Convert an input text (a statement) into a sequence of tokens. ... **### Step 2: Generate n-grams** From a list of tokens, ... **### Step 3: Count Overlapping n-grams** ... **### Step 4: Compute Modified n-gram Precision** ... **### Step 5: Calculate Brevity Penalty (BP)** ... **### Step 6: Combine Modified Precisions and BP to Compute the NLP Score** ...	... ### Final Results ### **[Step1]** : Tokens for a: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog."], Tokens for b: ... **[Step2]** : 2-gram for a: [('The', 'quick'), ('quick', 'brown'), ... **[Step3]** : ... **[Step4]** : Precision for 1-grams: 0.5556, Precision for 2-grams: 0.375, Precision for 3-grams: 0.2857, Precision for 4-grams: 0.1667 **[Step5]** : Brevity penalty (BP) = 1 **[Final]** : 0.3309	... ### Final Results ### **[Step1]** : Tokens for a: ["The", "quick", "brown", "fox", "jumps", "over", ("the",) "lazy", "dog", "."], Tokens for b: ... **[Step2]** : 2-gram for a: [('The', 'quick'), ('quick', 'brown'), ... **[Step3]** : ... **[Step4]** : Precision for 1-grams: 0.3333, Precision for 2-grams: 0.375, Precision for 3-grams: 0.2857, Precision for 4-grams: 0.1667 **[Step5]** : Brevity penalty (BP) = 1 ### Final Answer ### : The final answer is: 0.2777	... ### Step 5: Calculate the NLP Score ... ### NLP Score \text{NLP} = \\exp\\left(\\frac{1}{4}(\\log(0.5) + \\log(0.25) + \\log(0.125) + \\log(0.0625))\\right) \\]\n\\[\text{NLP} = \\exp\\left(\\frac{1}{4}(\\log(0.5) + \\log(0.25) + \\log(0.125) + \\log(0.0625))\\right) \\]\n\\[\text{NLP} = \\exp\\left(\\frac{1}{4}(\\log(0.5) + \\log(0.25) + \\log(0.125) + \\log(0.0625))\\right) \\]\n\\[\ ... (Results Not Generated)

Table 3: Detailed qualitative analyses. To demonstrate representative examples, we consolidate various error cases from Qwen2.5-7B into a single case for illustration.

that greedy decoding generally outperforms sampling methods for most evaluated tasks, we set the temperature to 0.0 to ensure deterministic outputs.

RQ1. Do current advanced LLMs possess the capabilities required by our benchmark? Table 2 presents the performance of various LLMs on the MCBench, and Figure 3 demonstrates FA for each metric. As evidenced by the experimental results, our benchmark challenges even the model used for data generation, gpt-4o, which achieves a final accuracy of only 40.98. This demonstrates that our benchmark demands a high level of complex instruction-following capability, underscoring the distinction between "knowing" a concept and "performing" it.

For reasoning models like QwQ, while their FA surpasses that of gpt-4o, their score on format following (FF) is noticeably lower, even lower than 7B-sized models. This suggests that although reasoning can improve task accuracy, it may significantly impair the ability to adhere to the format requirements of instructions. Therefore, enhancing performance on complex instructions, as required by our benchmark, necessitates competencies beyond mere reasoning enhancement.

Additionally, we observe distinct performance differences across categories. Despite the rubric requiring only a straightforward string matching algorithm, these variations indicate that LLMs have significant differences in handling various forms of input.

Table 3 displays the expected answers and error cases encountered during evaluation with MCBench. This highlights vulnerabilities in LLMs when dealing with complex instructions requiring mathematical reasoning. First, inadequacies in instruction handling can lead to incorrect outcomes starting from the tokenization phase. Even when tokenization is accurate, insufficient mathematical reasoning skills can result in incorrect calculations.

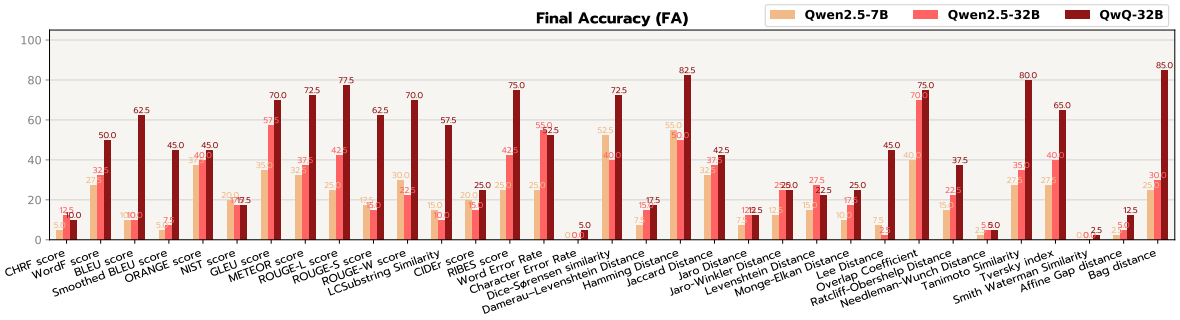

Figure 3: Final Accuracy (FA) score for each metric

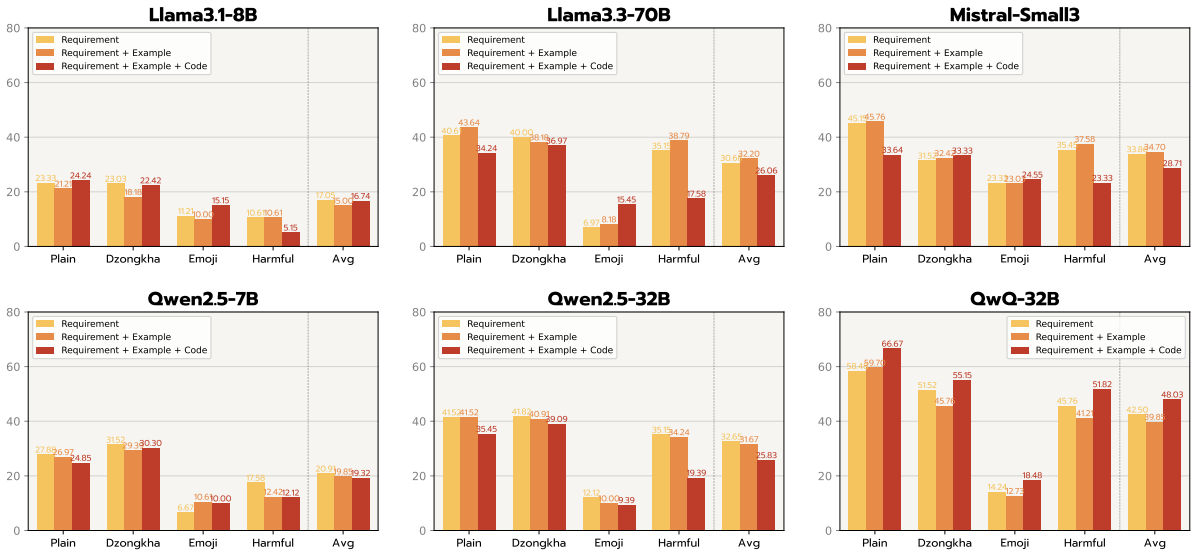

Figure 4: Performance differences based on the level of rubric details, which are reported using the FA metric

Additionally, while processing complex instructions, formatting instructions may be overlooked. In cases of inadequate long-range consistency, errors such as repeated incorrect phrases in lengthy text outputs may occur. These errors demonstrate that MCBench demands a comprehensive level of competence from LLMs.

RQ2. Does the amount of provided information impact instruction-following performance? In constructing a rubric for each metric, we included components such as requirements, examples, and code. We analyze how incorporating these elements affects performance. The experimental results are presented in Figure 4.

The findings indicate that more information does not necessarily lead to better performance. Notably, well-established and self-contained requirement statements alone allow current LLMs to demonstrate a reasonable degree of instruction-following capability, while the addition of code or examples does not consistently enhance performance. In the emoji category, incorporating code generally improved performance, whereas it led to general declines in the harmful category. This suggests that the required attributes vary based on input format, underscoring the effectiveness of our comprehensive benchmark in evaluating diverse scenarios.

Including code in the input proved beneficial for reasoning models but did not typically result in substantial performance gains overall. This might relate to the inherent code comprehension capabilities of LLMs. Models like QwQ, which perform deep input understanding, can achieve significant performance improvements with such detailed inputs, though other models often experienced performance declines.

RQ3. Is there a naming bias associated with using the term "NLP score" when constructing the rubric? There may be concerns that inherent familiarity with a specific metric name could negatively impact the abilities we aim to assess in our benchmark. To investigate the impact of pre-

Model	Original	NLP	Δ
Llama3.1-8B	15.45	**16.74**	-1.29
Qwen2.5-7B	**19.55**	19.32	+0.23
Mistral-Small3	**29.32**	28.71	+0.61
Qwen2.5-32B	**26.97**	25.83	+1.14
QwQ-32B	45.45	**48.03**	-2.58
Llama3.1-70B	**25.45**	24.55	+0.91
Llama3.3-70B	**26.21**	26.06	+0.15

Table 4: Performance difference between using established original metric names and employing the arbitrary term "NLP score" to describe the rubric

existing knowledge for each metric, we examine performance variations when using original metric names. The results, as shown in Table 4, indicate minimal differences between employing the original naming conventions and the newly introduced "NLP score." This suggests that our benchmark exhibits a certain robustness to intrinsic knowledge, and achieving high performance requires a sufficient capability in instruction understanding.

RQ4. Can specializing in Code/Math improve the abilities required by the benchmark? We investigate the impact of specialized capabilities on the performance of MCBench. To ensure a fair comparison, we evaluate the performance of the Qwen2.5-7B model released by the Qwen team alongside the concurrently released code and math models. The experimental results are presented in Table 5.

As the experimental results indicate, although there are some improvements in specific categories for the Code model, overall performance declines are observed in both the code and math specialized models. Notably, the math model shows a significant drop in SF, suggesting considerable decrease in instruction-handling capabilities, even its enhanced mathematical reasoning skills. For the Code model, while SF improves, we witness geneal decreases in FA, possibly due to a diminished ability to analyze and reason through given requirements.

These findings demonstrate that MCBench requires a comprehensive set of capabilities from LLMs. Enhancing a single aspect of performance yields limited benefits; achieving high scores generally necessitates strong performance across all capabilities. This underscores the robustness of our benchmark and its validity as an objective evaluation measure.

Category	Measure	Qwen	Qwen-Code	Qwen-Math
Requirements + Example + Code				
Plain	**FA**	24.85	**28.18** (+3.33)	20.91 (-3.94)
	FF	63.03	**69.09** (+6.06)	4.85 (-58.18)
	FD	**46.63**	45.76 (-0.87)	26.47 (-20.16)
Dzongkha	**FA**	**30.30**	19.7 (-10.6)	14.85 (-15.45)
	FF	**65.45**	60.3 (-5.15)	9.39 (-56.06)
	FD	**40.81**	39.49 (-1.32)	26.09 (-14.72)
Emoji	**FA**	10.00	**10.61** (+0.61)	8.48 (-1.52)
	FF	47.58	**63.64** (+16.06)	3.94 (-43.64)
	FD	30.89	**32.18** (+1.29)	26.21 (-4.68)
Harmful	**FA**	**12.12**	8.79 (-3.33)	8.48 (-3.64)
	FF	60.91	**67.27** (+6.36)	3.64 (-57.27)
	FD	**41.37**	37.01 (-4.36)	26.09 (-15.28)
Avg	**FA**	**19.32**	16.82 (-2.50)	13.18 (-6.14)
	FF	59.24	**65.08** (+5.84)	5.45 (-53.79)
	FD	**39.92**	38.61 (-1.31)	26.21 (-13.71)
Requirements Only				
Plain	**FA**	**27.88**	24.85 (-3.03)	16.06 (-11.82)
	FF	60.91	**84.24** (+23.33)	7.58 (-53.33)
	FD	39.78	**41.8** (+2.02)	26.15 (-13.63)
Dzongkha	**FA**	**31.52**	20.61 (-10.91)	10 (-21.52)
	FF	73.33	**77.88** (+4.55)	7.58 (-65.75)
	FD	39.58	35.57 (-4.01)	26.09 (-13.49)
Emoji	**FA**	6.67	**6.97** (+0.30)	**6.97** (+0.30)
	FF	52.73	**66.97** (+14.24)	3.03 (-49.7)
	FD	30.65	**30.88** (+0.23)	26.03 (-4.62)
Harmful	**FA**	**17.58**	13.03 (-4.55)	4.24 (-13.34)
	FF	58.48	**68.18** (+9.70)	9.70 (-48.78)
	FD	34.33	**34.53** (+0.20)	26.19 (-8.14)
Avg	**FA**	**20.91**	16.36 (-4.55)	9.32 (-11.59)
	FF	61.36	**74.32** (+12.96)	6.97 (-54.39)
	FD	**36.08**	35.70 (-0.38)	26.11 (-9.97)

Table 5: Performance differences between the code-specialized model and the math-specialized model are evaluated. All models report their performance based on Qwen2.5-7B.

5 Conclusion

This paper introduces **MCBench**, a comprehensive and objective benchmark designed to evaluate LLMs. **MCBench** consists of a step-by-step rubric for executing a string matching metric, candidates for metric calculation, and reference code to objectively assess LLM outputs. By ensuring LLMs strictly follow the provided rubric to compute metrics, we comprehensively evaluated three attributes: complex instruction following, mathematical reasoning, and long-range consistency. We expanded our analysis by introducing four candidate categories, three variants of the rubric components, and three distinct evaluation measures. Our benchmark revealed that even advanced LLMs, such as GPT-4o, achieved only a performance level of 40.98. Enhancing a singular capability, such as code or math specialization, proved minimally effective. Through these analysis, we demonstrated that MCBench is a highly effective objective bench-

mark for thoroughly assessing LLM capabilities. In future research, we plan to introduce an objective benchmark that incorporates tool implementation.

Limitation

While MCBench targets string-matching metrics, other NLP primitives—e.g., probabilistic measures, graph-based scores, or differentiable similarity functions—remain unexplored. Extending the benchmark to these domains will broaden its coverage. To isolate the LLM's ability to follow complex instructions, we deliberately excluded external tools such as Python implementations in evaluating LLM. This approach allowed us to elicit the inherent mathematical reasoning capabilities of the LLM.

Ethics Statement

We conducted a human inspection of all generated data to ensure there were no ethical issues. The harmful text we used was sourced from the dataset released by (Hui et al., 2024b), and it does not contain inherent ethical problems. However, if the prompt leads to the generation of harmful responses, it could pose ethical concerns. It is important to note that the purpose of using such data is to assess whether the LLM overreacts to harmful text. We strongly oppose any attempts to solicit responses to these prompts. An AI assistant contributed to the writing of this paper by providing grammar checking and writing support only. The assistant did not contribute to the research content or the development of the study's topic.

Acknowledgements

This work was partly supported by ICT Creative Consilience Program through the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (IITP-2025-RS-2020-II201819, 25%) (RS-2024-00398115, 25%) (No. RS-2022-II220369, (Part 4) Development of AI Technology to support Expert Decision-making that can Explain the Reasons/Grounds for Judgment Results based on Expert Knowledge, 25%) and grant funded by Institute of Information & communications Technology Planning & Evaluation(IITP) under the Leading Generative AI Human Resources Development(IITP-2025-R2408111, 25%) grant funded by the Korea government(MSIT).

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–718, Nusa Dua, Bali. Association for Computational Linguistics.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level BLEU. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA. Association for Computational Linguistics.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or LLMs as the judge? a study on judgement bias. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8301–8327, Miami, Florida, USA. Association for Computational Linguistics.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176.

Michel-Marie Deza and Elena Deza. 2006. *Dictionary of distances*. Elsevier.

Lee R. Dice. 1945. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Florian E. Dorner, Vivian Yvonne Nastl, and Moritz Hardt. 2025. Limits to scalable evaluation at the frontier: LLM as judge won't beat twice the data. In *The Thirteenth International Conference on Learning Representations*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pages 30039–30069. Curran Associates, Inc.

Mingqi Gao, Xinyu Hu, Xunjian Yin, Jie Ruan, Xiao Pu, and Xiaojun Wan. 2025. Llm-based nlg evaluation: Current status and challenges. *Computational Linguistics*, pages 1–28.

Osamu Gotoh. 1982. An improved algorithm for matching biological sequences. *Journal of molecular biology*, 162(3):705–708.

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

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, et al. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell system technical journal*, 29(2):147–160.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2024. Human feedback is not gold standard. In *The Twelfth International Conference on Learning Representations*.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. 2024a. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*.

Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. 2024b. Pleak: Prompt leaking attacks against large language model applications. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3600–3614.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, MA. Association for Computational Linguistics.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Matthew A Jaro. 1989. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical association*, 84(406):414–420.

Mehran Kazemi, Bahare Fatemi, Hritik Bansal, John Palowitch, Chrysovalantis Anastasiou, Sanket Vaibhav Mehta, Lalit K Jain, Virginia Aglietti, Disha Jindal, Peter Chen, et al. 2025. Big-bench extra hard. *arXiv preprint arXiv:2502.19187*.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 501–507, Geneva, Switzerland. COLING.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Mistral. 2025. Mistral small 3. https://mistral.ai/news/mistral-small-3. Published 2025-01-30.

Alvaro E Monge, Charles Elkan, et al. 1996. The field matching problem: algorithms and applications. In *Kdd*, volume 2, pages 267–270.

Hyeonseok Moon, Jaehyung Seo, Seungyoon Lee, Chanjun Park, and Heuiseok Lim. 2025. Find the intention of instruction: Comprehensive evaluation of instruction understanding for large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5944–5964, Albuquerque, New Mexico. Association for Computational Linguistics.

Andrew Morris, Viktoria Maier, and Phil Green. 2004. From wer and ril to mer and wil: improved evaluation measures for connected speech recognition.

Allan H Murphy. 1996. The finley affair: A signal event in the history of forecast verification. *Weather and forecasting*, 11(1):3–20.

Andrew Mutton, Mark Dras, Stephen Wan, and Robert Dale. 2007. Gleu: Automatic evaluation of sentence-level fluency. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 344–351.

Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

OpenAI. 2025. Openai o3-mini system card. https://openai.com/index/o3-mini-system-card. Published 2025-01-31.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. Language model tokenizers introduce unfairness between languages. In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Maja Popović. 2017. chrF++: words helping character n-grams. In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, et al. 2025. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *arXiv preprint arXiv:2504.16074*.

Qwen. 2025. Qwq-32b: Embracing the power of reinforcement learning.

John W. Ratcliff and David E. Metzener. 1988. Pattern matching: The gestalt approach. *Dr. Dobb's Journal of Software Tools*, 13(7):46–51, 68–72. Accessed 24 Apr 2025.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685.

Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. 2024. Judging the judges: A systematic investigation of position bias in pairwise comparative assessments by llms. *arXiv preprint arXiv:2406.07791*.

George Gaylord Simpson. 1943. Mammals and the nature of continents. *American Journal of Science*, 241(1):1–31.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Temple F Smith, Michael S Waterman, et al. 1981. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197.

Yifan Song, Guoyin Wang, Sujian Li, and Bill Yuchen Lin. 2025. The good, the bad, and the greedy: Evaluation of LLMs should not ignore non-determinism. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4195–4206, Albuquerque, New Mexico. Association for Computational Linguistics.

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. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. Featured Certification.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Taffee T Tanimoto. 1958. *An elementary mathematical theory of classification and prediction*. International Business Machines Corporation.

Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Peter Weiner. 1973. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking complex instruction-following with multiple constraints composition. In *Advances in Neural Information Processing Systems*, volume 37, pages 137610–137645. Curran Associates, Inc.

William E Winkler. 1990. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

JP Woodard and JT Nelson. 1982. An information theoretic measure of speech recognition performance. In *Workshop on standardisation for speech I/O technology, Naval Air Development Center, Warminster, PA*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2024a. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024b. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Xiaosen Zheng, Tianyu Pang, Chao Du, Qian Liu, Jing Jiang, and Min Lin. 2025. Cheating automatic LLM benchmarks: Null models achieve high win rates. In *The Thirteenth International Conference on Learning Representations*.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

Caleb Ziems, William Held, Omar Shaikh, Jiaao Chen, Zhehao Zhang, and Diyi Yang. 2024. Can large language models transform computational social science? *Computational Linguistics*, 50(1):237–291.

A Dataset Details

We report the data statistics of MCBench in Table 6.

Basic Statistics	
# Rubric	33
# Candidates	40
# Test Instances	1,320
Avg/Min/Max # Steps	4.18 / 3 / 7
Avg # character - Rubric	
Requirement	2259.3
Requirement + Example	3096.2
Requirement + Example + Code	5145.0
Avg # character - Candidate	
Plain	24.3
Dzongkha	31.2
Emoji	16.4
Harmful	46.3

Table 6: Data Statistics.

To determine candidates, we organized data such that character lengths were similar across categories to minimize difficulty differences. We avoided constructing samples with excessively long character lengths, as processing such candidates would also take longer.

The character length of plain candidates served as the baseline. To generate these plain candidates, we prompted GPT-4o with the following query: *"Recommend several examples that are appropriate to measure..."*. We assumed the character lengths of these generated candidates as standard. We ensured that the data for Dzongkha, Emoji, and Harmful categories did not significantly deviate from the length of these plain candidates.

For Dzongkha and Emoji, we used prompts such as, *"Recommend syntactically similar sentence pairs written in {Emoji, Dzongkha}"* to generate data. Initially, candidates were generated using these prompts, then the authors select and adjust the data to meet length criteria. We utilized GPT-4o throughout this data generation process.

For the Harmful data category, we adopted the dataset proposed by (Hui et al., 2024b) to construct our benchmark. We focused on the shortest data instances to consider character length. However, this data tends to be twice as long as plain candidates, which might introduce inherent difficulty.

B Evaluation Criteria for Human Evaluation

For the **Requirement**, we focus on verifying that the descriptions do not contradict the reference documentation. We design our requirements by combining purposefully generated content from three models. If a description is inaccurate or requires additional explanations for a step, we regenerate the requirement for that step to ensure quality. Crucially, we confirm the completeness by verifying that the requirement is self-contained, allowing the metric to be computed directly from the description alone.

For **Example** and **Code**, we focus on verifying that they do not contradict the requirements. In verifying **Example**, we ensure they accurately illustrate each step with concise execution instances. Similarly, for **Code**, we check that each step is appropriately addressed. Specifically, the code undergoes actual Python implementation, and if any execution errors arise, the authors directly correct them to ensure functionality.

C Model Details

Table 7 shows several LLMs we employed for our experiments and data construction. By conducting experiments with LLMs of varying performance levels, we aim to verify the robustness of our benchmark.

D Evaluation Details

To evaluate the performance of MCBench, we provided each model with the prompts as shown in Table 8. In this prompt, **Statement A** and **Statement B** refer to the two sentence pairs that comprise the **Candidates**.

E Data Construction Details

The prompt we used for data construction is shown in Table 9.

F Justification for Selecting String-Match Metrics

Our benchmark primarily focuses on string-matching metrics for several reasons:

Model Name	# Params
Llama3.1-8B (Grattafiori et al., 2024) : `meta-llama/Meta-Llama-3.1-8B-Instruct`	8.03B
Llama3.1-70B (Grattafiori et al., 2024) : `meta-llama/Llama-3.1-70B-Instruct`	70.6B
Llama3.3-70B (Grattafiori et al., 2024) : `meta-llama/Llama-3.3-70B-Instruct`	70.6B
Mistral-Small3 (Mistral, 2025) : `mistralai/Mistral-Small-24B-Instruct-2501`	23.6B
Qwen2.5-7B (Yang et al., 2024a) : `Qwen/Qwen2.5-7B-Instruct`	7.62B
Qwen2.5-32B (Yang et al., 2024a) : `Qwen/Qwen2.5-32B-Instruct`	32.8B
QwQ-32B (Qwen, 2025) : `Qwen/QwQ-32B`	32.8B
Llama3.3-70B(R1) (Guo et al., 2025) : `deepseek-ai/DeepSeek-R1-Distill-Llama-70B`	70.6B
Qwen2.5-32B(R1) (Guo et al., 2025) : `deepseek-ai/DeepSeek-R1-Distill-Qwen-32B`	32.8B
Qwen2.5-Coder-7B (Hui et al., 2024a) : `Qwen/Qwen2.5-Coder-7B-Instruct`	7.62B
Qwen2.5-Math-7B (Yang et al., 2024b) : `Qwen/Qwen2.5-Math-7B-Instruct`	7.62B
gpt-4o-mini (Hurst et al., 2024) : `gpt-4o-mini-2024-07-18`	-
gpt-4o (Hurst et al., 2024) : `gpt-4o-2024-08-06`	-
o1 (Jaech et al., 2024) : `o1-2024-12-17`	-
o3-mini (OpenAI, 2025) : `o3-mini-2025-01-31`	-

Table 7: Model Details. We deployed OPENAI API call for experiments with GPT-4o-mini and GPT-4o, and `huggingface` (Wolf et al., 2020) for eliciting model weights for other publicly-available LLMs.

- It does not require contextual understanding of "candidate". This approach isolates the ability to "understand and execute instructions" from the ability to "process given inputs." This allows us to analyze the limitations of LLMs' inherent input understanding by evaluating performance variations based on different type of candidates. Defining metrics enables constructing parallel reference code and objective evaluation of scores. Clearly defined outputs allow for objective assessments, which is a significant advantage of our benchmark.

- It effectively evaluates long-range consistency. This "long-range"includes not only inputs composed of multi-step instructions but also extended solutions required to derive the final answer. The benchmark is designed to assess LLMs' ability to maintain consistency, follow instructions, and perform numerical reasoning for accurate outcomes with long input-output interactions.

- It comprehensively evaluates mathematical reasoning and instruction following. Unlike benchmarks like AIME or MATH-500, where the LLM must "reason" to find a path for solutions, this benchmark requires intact adherence to given instructions while also demonstrating mathematical skills. While data sorting and financial calculations could be considered under this purpose, they seem beyond our benchmark's defined scope. A broader definition would be necessary to encompass these tasks, but we view that presenting such an objective and experimental scope is slightly beyond our current focus.

We find it impractical to consider all code-implementable measures, as indiscriminate inclusion could obscure the benchmark's purpose. While future research may pursue a more comprehensive definition, we believe focusing on our initial scope is preferable. Therefore, we clearly define our objectives and systematically present findings within that scope.

System Prompt

You are an expert in computer science and a highly capable, responsive assistant trained to assist with a broad range of tasks.
I have derived a new metric, ***{NLP score}***.
{NLP score} is an automatic evaluation metric used for comparing the similarity between a hypothesis and reference text.
I will show you the detailed process to calculate this metric.
Your task is to derive the ***{NLP score}*** for the given input values by referring to the provided process.
At each intermediate step, print the corresponding result.
Before generating the final answer, briefly explain your reasoning behind it. You must find your final answer before your response reaches 1000 lines. Finally, compile all the results and output your final verdict by strictly following this format:

Final Results
[Step1] : ...
[StepN] : ...
[Final] : ...

User Query

{Rubric}

Tell me the ***{NLP score}*** of "a" with a reference text "b", following the previous rubric:
a = ***{Statement A}***
b = ***{Statement B}***

Table 8: Prompt for evaluating LLMs on MCBench

System Prompt
You are an expert in computer science and a highly capable, responsive assistant trained to assist with a broad range of tasks. Your main goal is to provide helpful, relevant, and accurate information to users. Your responses should be clear and adaptable to different levels of user expertise. Be concise but detail-oriented.
User Query
I want to calculate **{*Metric Name*}** between statement A and B. Explain necessary detailed procedures step-by-step. Please explain the processes required at each step in a clear and detailed manner. Please separate each step with "### Step N". Your explanation for each step must include the following three parts: "Requirement", "Example", and "Code". Detailed instructions are as follows: - Requirement Part You must explain detailed procedures required for each step. Descriptions should be self-contained that anyone can re-implement it without referring other documents. Your explanation should be concise but contain sufficient details. You must make a response in a definitive way. If necessary, separate subprocesses for each step. Requirement part should not contain any code or examples. If formulas are necessary, output them in LaTeX format. Begin your descriptions with "[Requirement]". - Example Part For each step, you must include a simple implementation example. Provided example should be aligned with Requirement. Begin your descriptions with "[Example]". - Code Part You must provide Python code for each step in a way that it can be executed without any packages, implemented as functions. Name of each function should be "stepN" that corresponds to each step. After providing your function code, give me an example usage of it. Provided example should be aligned with Requirement. Begin your descriptions with "[Python Code]".

Table 9: Prompt for constructing MCBench. Note that every data generated with this prompt were later refined through human inspection.