Classifying and rating AI benchmarks

Reece Shiraishi

Fermi National Accelerator Laboratory, Batavia, Illinois 60510

Abstract

We created a set of standards to efficiently evaluate AI benchmarks through objective means. Although prevalent, especially in recent times, AI benchmarks have no single way to measure their effectiveness. The MLCommons team provided a set of criteria for evaluating benchmarks, although the criteria lacks a clearly defined set of evaluation rules. We created a rubric with preset factors to efficiently and objectively evaluate a benchmark's quality. We created a software framework for processing lists of benchmarks for visualization. The framework and rating system allows researchers to quickly check if their benchmarks are effective.

Introduction

In modern times, AI tools are more prevalent than ever. Most web browsers, schools, and scientific labs use AI on a daily basis. Despite the rapid rise in AI technology, systematic methods of measuring their performance are not as widely known. Several researchers published benchmarks to quantify model performance, but the benchmarks vary in quality. Most measure different aspects of a model, making quality judgments harder.

Despite the rapidly expanding projects on AI benchmarking, there is no universally accepted way to exactly determine the quality of a benchmark. The term “AI benchmark” is poorly defined and understood, further complicating precise judgments.

This project seeks to provide criteria for determining benchmark quality. Researchers will have the ability to compare their own benchmarks to others using a single definition and rating scale. Given only the ratings, research teams will be able to easily select the best benchmarks for their use cases, improving the efficiency of future research.

Background

AI research has exploded in recent times. There was a sudden increase in computer science publications around 2010, a time widely recognized as a turning point in AI research [1]. With the rise of large language models in the early 2020s, effective and accessible AI that can sometimes outperform humans entered the consumer market.

An increase in AI research brought with it a rise in ways to measure AI performance. Vast numbers of benchmarks appeared in literature, measuring tasks such as using common-sense reasoning, ability to perform scientific research, and solving fluid dynamics problems. AI models targeted for science could be expected to solve math problems, control sensors, match patterns, or check if a hypothesis is confirmed [2].

Despite the rise in available benchmarks, a commonly accepted definition of “AI benchmark” does not exist. One paper states that benchmarks are defined by problem class, setting, and cases, but states that ways of measurement may be defined differently [2]. Another article stated criteria on whether an AI model is beneficial to humanity [3], despite lacking a general definition.

Other than the two articles cited, [2] and [3], a search on Google Scholar for “ai benchmark definition” and “ai benchmark classification” yielded no relevant results. Searches yielded benchmarks (not definitions) and whether the benchmarks should be trusted.

The work covered in this paper hopes to facilitate the creation of high-quality, easily verifiable benchmarks. My work includes criteria for rating, objective rubrics for each criterion, and the ability for others to view and process existing benchmarks, with emphasis on rubrics and a viewing system.

Methods

Prior to work, my supervisor’s team developed a series of metrics for evaluating AI benchmarks. Their criteria include: clear problem specification, easily accessible datasets, quantifiable measures of comparison, provided reference solutions, and good documentation to reproduce the results.

The team previously used AI tools to compile a list of 23 benchmarks. The LLMs used were relatively accurate and extremely fast, but the output still required manual checking. Prior to starting work, the team that initially created the list warned that AI can sometimes fabricate its output. To ensure the presence and accuracy of the list, each compiled benchmark was manually verified to check the AI’s work. Some of the model’s benchmarks in its final list contained standalone models instead of a benchmark. One benchmark’s provided link led to a 404 Not Found error.

At the time, the definitions for benchmark rating were not yet finalized. The list of benchmarks was converted to a machine-parseable YAML format to enable faster analysis. The fields listed in each YAML entry are outlined in Appendix A.

An extra entry, “ratings”, was added later to hold the benchmark’s quality scores.

A Python script was written to convert the YAML file into a Markdown or TeX table. Each entry in the YAML became a row in the output table (Appendix B). Depending on command-line flags, the Python script’s user can specify the input files, output directory, and table format. In TeX format, the user can also specify whether to produce a standalone document with a header. Both the Python script and the YAML files were posted to the team’s Github site, where additional instructions for users were added in Readme files.

To make its output easier to read, the Python script had more command-line flags added to it. The first option added was one to remove the ‘cite’ row from the Markdown table. BibTeX citations are complicated and may take lots of space. One of the citations had over 100 authors, taking as much space as all the other entries combined. With less space used, the table was more readable and easier to make quick judgments. The input files, output directory, and table format became mandatory arguments to prevent future sources of problems. Users may forget to add the arguments, forcing the program to guess and introducing a possible source of bugs.

For easier use, the program included a Makefile. Entering the many command-line flags each time the program runs proved tedious, often causing user errors. Some flags required that other flags were used, leading to complicated rules. The Makefile enabled users to use a single command in place of long chains of command-line flags.

The most critical part of the project was rating each benchmark based on how well it matches the preset definition. As soon as they could, the benchmarking team finalized definitions for the 5 criteria for judging the quality of a benchmark.

To properly compare benchmarks, a rating system based on the given criteria was created. Each criterion was rated on a scale from 0 to 10, with 10 being the maximum score. A category could receive a score of N/A if its score cannot be accurately determined, or required information is inaccessible. Criteria for evaluation are in Appendix C.

Key points about the 23 benchmarks, along with 45 additional benchmarks, were translated to YAML format. Each listing included information required to find and use the benchmark, such as the benchmark's name, a URL to the benchmark, BibTeX citation entries, and whether the benchmark is deprecated. A "ratings" section containing the rubric scores was added to the end of each YAML entry. Explanations for each of the five criteria accompanied the numerical ratings.

Manually rating each of the over 60 benchmarks would be tedious and highly subjective, so an LLM tool was used for initial ratings. Starting with the highest rated benchmarks, each output was manually verified. Most evaluations turned out to be too lenient, especially in the "reproducible protocol" section, where most highly rated benchmarks had insufficient information to find the results.

The project's Python script underwent heavy modification as a result to handle the ratings. Because of the way YAMLs are parsed in Python, each change to the YAML format required a change in the Python script. Each rating, along with a reason for giving the rating, was appended to the columns of the output tables. The modifications worked well, but were inflexible to future changes. The definition and rating system could be changed later, but any change, such as adding an extra criterion, would cause problems in the script.

The team also decided to change the format of the input YAML files, making the current script unusable on the new format. For better ease of use, each column in the new YAML format

included a “required” or “optional” field, stating whether the field must be included in the final table. The columns also included a definition containing the meaning of the column, resolving any ambiguity on what should be placed inside.

The Python script was further modified to accommodate the descriptions and conditions. The description and condition of each entry was omitted to preserve the table’s current formatting. For easier maintenance, each section of the script was divided into separate modules. At about 1000 lines long and containing almost no documentation, the script appeared almost unreadable. Some functionality had not been used, while other seemingly useless pieces were crucial. The code refactoring made future development easier and faster.

Results

The project produced systematic and quantifiable methods for classifying benchmarks, along with a converter into machine- and human-readable formats. Other researchers can easily compare new benchmarks against the project’s compiled list and accurately determine benchmark quality. The benchmark rating criteria enables better measurements of AI performance, accommodating rapid AI advances seen in research and everyday use.

Acknowledgements

Special thanks to my partner Anjay Krishnan and my supervisors Ben Hawks, Jovan Mitrevski, and Gregor von Laszewski.

This manuscript has been authored by Fermi Forward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI).

Appendix A: Detailed descriptions of each criterion

Clear problem specification: Describes “the type of input data, the expected output, and any constraints on the task”. Precisely describes the model’s inputs and outputs, including an overview of how the model should produce its output. Constraints may include hardware, power consumption, or carbon footprint.

Datasets: Adheres to FAIR (findability, accessibility, interoperability, reproducibility of results) principles. The benchmark should provide all necessary tools and programs to reproduce any given result. Data may be split into a training, testing, and validation set for the evaluated model to use.

Measures of comparison: Contains a quantifiable measure of model performance. Examples include the number of correct answers or mean squared error from the correct result. Hardware comparison, such as the number of watts consumed while running the benchmark, may also be considered.

Reference solutions: At least one model that ran the benchmark and has posted results. Papers about benchmarks often include performance by commonly used models, such as ChatGPT and DeepSeek.

Documentation: Gives users the ability to reproduce the reference solution. Common techniques include setup programs, additional instructions, and runtime environments for all necessary software libraries.

Appendix B: Some field titles and their associated description in the YAML format

Field Title	Description
date	Date of publication
expired	Date of the benchmark's retirement. `null` if the benchmark is active.
name	Title of the benchmark
url	Web link to the benchmark's associated paper
domain	Field of scientific study (i.e. mathematics, biology)
focus	One sentence description of what the benchmark tests
keywords	Keywords listed by the benchmark's paper, or important words mentioned in the benchmark
description	More detailed summary of the benchmark's tasks
task_types	What the model does
ai_capability_measured	Abilities that the model is evaluated on
metrics	Method of comparing models to one another

models	Notable AI models tested using the benchmark
notes	Any additional notes that the benchmark compiler wishes to state, i.e. “Not a benchmark”, “Exact metrics will be clarified soon”
cite	Citation in BibTeX format

Appendix C: Initial rubric for evaluating benchmarks

1. Problem Specification & Constraint: Clarity of task, inputs, outputs, and system constraints.

- 9–10: Clear task, inputs/outputs defined and format specified, system constraints quantified (e.g., latency, hardware).
- 7–8: Well-defined task, most constraints present / Slight lack of clarity about either inputs or outputs
- 5–6: Moderately well defined task / critical information about inputs and outputs (i.e. question type, format) are unclear or missing, but most other info is present
- 3–4: Vague task description, inputs or outputs not specified.
- 1–2: Task intent unclear.
- 0: Not present

2. Dataset (FAIR Principles: Findability, Accessibility, Interoperability, Reusability; versioned,

stable splits.

- 9–10: Fully FAIR, documented, versioned, reproducible with train/test/validation set splits.
- 7–8: Mostly FAIR, good documentation but minor gaps. At most 1 of the FAIR principles missing or questionable.
- 5–6: Usable dataset, but not fully FAIR or missing metadata/versioning. <=2 FAIR principles missing or questionable.
- 3–4: Partially accessible or inconsistently formatted. <=3 FAIR principles missing or questionable.
- 1–2: Closed or unstandardized data. All FAIR principles missing or questionable.
- 0: No dataset available

3. Performance Metrics: Quantifiable measures for comparison

- 9–10: Methods for comparison quantitative. Measured model's output entirely falls within the stated comparison method. Reviewers do not need to guess the meaning of the benchmark results.
- 7–8: Some subjective measures / small portions of the model's output may not be covered by the metrics / reviewer must take minor inferences to interpret the results
- 5–6: Mostly subjective measures / metrics do not cover significant portions of model output / output requires substantial guesswork to interpret
- 3–4: Nearly all subjective measures / model output is mostly outside of the metrics / meaning of the output is poorly specified
- 1–2: Metrics do not explain model output / results are extremely difficult to quantify or

measure

- 0: No metrics stated

4. Reference Solution: Baseline implementation demonstrating task feasibility.

- 9–10: Well documented, reproducible baseline provided with performance results.
- 7–8: Functional baseline included and evaluated, but lacks full documentation or reproducibility details.
- 5–6: Baseline exists but is not evaluated using the benchmark or has little instructions for reproduction
- 3–4: Baseline performance briefly mentioned but not run with the benchmark, or partial implementation with no reported results.
- 1–2: Reference solution briefly mentioned but is vague
- 0: No reference solution or baseline presented

5. Reproducible Protocol: Code, environment, and instructions to reproduce results.

- 9–10: Full code, environments, and scripts. Any resources that are not provided have easily followed instructions to obtain and use them.
- 7–8: Code provided but environment setup or instructions incomplete.
- 5–6: Partial code or missing critical environment setup details.
- 3–4: Code inaccessible or incomplete. Environment setup details missing.
- 1–2: Missing 2 of the 3 of: accessible code, environment install instructions, use instructions
- 0: No Code, environment or instructions to reproduce results provided

References

[1]V. Carchiolo and M. Malgeri, Science and Technology Publications, **577** (2024).

[2]Y. Li and J. Zhan, ScienceDirect 2, **100063** (2022).

[3]J. Cowls and A. Tsamados and M. Taddeo and L. Floridi, Nat Mach Intell 3, **111** (2021).