

## A NeurIPS Datasets and Benchmark Track Dataset Documentation

### A.1 Links

All items can be found in this collection: <https://hf.co/collections/allenai/reward-bench-65e8dea22eb775e2e7d5dbe2>.

- Core dataset on HuggingFace: <https://hf.co/datasets/allenai/reward-bench>
  - Dataset Croissant metadata: <https://hf.co/datasets/allenai/reward-bench/blob/d61ac5c97d0f2f7eb922da3c0b6bbaf614ae8d02/README.md?code=true#L2>
  - Dataset DOI identifier (10.57967/hf/2457): <https://huggingface.co/datasets/allenai/reward-bench?doi=true>.
  - Reformatted prior test sets: <https://hf.co/datasets/allenai/preference-test-sets>
  - Evaluation results data: <https://huggingface.co/datasets/allenai/reward-bench-results>
  - Script to re-build the dataset from scratch: [https://huggingface.co/datasets/allenai/reward-bench/blob/main/build\\_datasets.ipynb](https://huggingface.co/datasets/allenai/reward-bench/blob/main/build_datasets.ipynb)
  - Dataset license (ODC-by): <https://huggingface.co/datasets/allenai/reward-bench/blob/main/LICENSE.md>. **The authors bear all risks in the case of violation of this dataset license.** In our case, ODC-by is used because most of the dataset is repurposing other data, rather than original generations.
- Codebase: <https://github.com/allenai/reward-bench>
- Leaderboard: <https://hf.co/spaces/allenai/reward-bench>

### A.2 Using this dataset

This dataset can be used with any codebase that can read parquet records. The most popular for doing so, which is used in our codebase on GitHub, is HuggingFace Datasets. To load the dataset minimally with Datasets:

```
# pip install datasets
from datasets import load_dataset
data = load_dataset("allenai/reward-bench", split="filtered")
```

Note that ‘split=raw’ shows the samples before manual verification. A dataset viewer is available in the Leaderboard and natively on HuggingFace’s dataset pages.

**This dataset will be maintained indefinitely. Any changes to the dataset location will automatically be redirected by HuggingFace. Future improvements will be handled as new datasets or splits, e.g. ‘filtered-v2’**

### A.3 Benchmark Reproducibility

To exactly reproduce the results of the benchmark, we include a Dockerfile (<https://github.com/allenai/reward-bench/blob/main/Dockerfile>) and configurations for every model we evaluated ([https://github.com/allenai/reward-bench/blob/main/scripts/configs/eval\\_configs.yaml](https://github.com/allenai/reward-bench/blob/main/scripts/configs/eval_configs.yaml)). All experiments are conducted on NVIDIA A100 GPUs.

We plan to host the dataset indefinitely on HuggingFace.

### A.4 Dataset Card

**Note: this is the dataset is card converted from HuggingFace.**

The RewardBench evaluation dataset evaluates capabilities of reward models over the following categories:

1. **Chat:** Includes the easy chat subsets (alpacaeval-easy, alpacaeval-length, alpacaeval-hard, mt-bench-easy, mt-bench-medium)
2. **Chat Hard:** Includes the hard chat subsets (mt-bench-hard, llmbar-natural, llmbar-adver-neighbor, llmbar-adver-GPTInst, llmbar-adver-GPTOut, llmbar-adver-manual)
3. **Safety:** Includes the safety subsets (refusals-dangerous, refusals-offensive, xstest-should-refuse, xstest-should-respond, do not answer)
4. **Reasoning:** Includes the code and math subsets (math-prm, hep-cpp, hep-go, hep-java, hep-js, hep-python, hep-rust)

The RewardBench leaderboard averages over these subsets and a final category from [prior preference data test sets](#) including Anthropic Helpful, Anthropic HHH in BIG-Bench, Stanford Human Preferences (SHP), and OpenAI’s Learning to Summarize data.

The scoring for RewardBench compares the score of a prompt-chosen pair to a prompt-rejected pair. Success is when the chosen score is higher than rejected.

In order to create a representative, single evaluation score, we perform a limited mixture of averaging across results. For all the subsets detailed below except for Reasoning, we perform per-prompt weighted averaging across all the prompts in the subset to get the section score. For example, in Chat we take a weighted average of the AlpacaEval and MT Bench sets based on the number of prompts. For Reasoning, we increase the weight of the PRM-Math subset so code and math abilities are weighed equally in the final number, rather than increasing the relevance of code. Once all subsets weighted averages are achieved, the final RewardBench score is the average across the subset scores (including Prior Sets).

#### A.4.1 Dataset Details

In order to maintain all the relevant data, the samples in the dataset will have the following items. Note, the dataset is single-turn: \* `prompt (str)`: the instruction given in the various test sets. \* `chosen (str)`: the response from the better model or the better rated prompt. \* `chosen_model (str)`: where applicable \* `rejected (str)`: the response with the lower score or from word model. \* `rejected_model (str)`: where applicable \* `subset (str)`: the subset (e.g. alpacaeval-easy) of the associated prompt as the dataset is all in one split. \* `id (int)`: an incremented id for every prompt in the benchmark.

To select a specific subset use HuggingFace Datasets `.filter` functionality.

```
dataset = dataset.filter(lambda ex: ex["subset"] == "alpacaeval-easy")
```

This can easily be converted to the standard chosen/rejected list of messages format (see [UltraFeedback for an example](#)), for example with our data loading utilities on [GitHub](#).

**Subset Summary** Total number of the prompts is: 2985.

Subset	Num. Samples (Pre-filtering, post-filtering)	Description
alpacaeval-easy	805, 100	Great model vs poor model; GPT4-Turbo 97.7% v. Alpaca 7b 26.46% (data <a href="#">here</a> )
alpacaeval-length	805, 95	Good model vs low model, similar length; Llama2chat 70B 92.66% vs Guanaco 13B 52.61% (data <a href="#">here</a> )
alpacaeval-hard	805, 95	Great model vs baseline model; Tulu 2 95.0% v. Davinici003 50.0% (data <a href="#">here</a> )
mt-bench-easy	28, 28	MT Bench 10s vs 1s (source <a href="#">data</a> )
mt-bench-medium	45, 40	MT Bench 9s vs 2-5s (source <a href="#">data</a> )
mt-bench-hard	45, 37	MT Bench 7-8 vs 5-6 (source <a href="#">data</a> )

Subset	Num. Samples (Pre-filtering, post-filtering)	Description
refusals-dangerous	505, 100	Dangerous rejected response vs polite chosen refusal
refusals-offensive	704, 100	Offensive rejected response vs polite chosen refusal
llmbar-natural	100	Manually curated instruction pairs (See <a href="#">paper</a> )
llmbar-adver-neighbor	134	Adversarial instruction response vs. off-topic prompt response (See <a href="#">paper</a> )
llmbar-adver-GPTInst	92	Adversarial instruction response vs. GPT4 generated off-topic prompt response (See <a href="#">paper</a> )
llmbar-adver-GPTOut	47	Adversarial instruction response vs. unhelpful-prompted GPT4 responses (See <a href="#">paper</a> )
llmbar-adver-manual	46	Challenge set manually designed chosen vs. rejected
xstest-should-refuse	450, 250	False response dataset (see <a href="#">paper</a> )
xstest-should-respond	450, 154	False refusal dataset (see <a href="#">paper</a> )
do not answer	939, 136	<a href="#">Prompts which responsible LLMs do not answer</a> : Refusals are chosen and responses are rejected
hep-cpp	164	C++ working code vs. buggy code (See <a href="#">dataset</a> or <a href="#">paper</a> )
hep-go	164	Go working code vs. buggy code
hep-java	164	Java working code vs. buggy code
hep-js	164	Javascript working code vs. buggy code
hep-python	164	Python working code vs. buggy code
hep-rust	164	Rust working code vs. buggy code
math-prm	447	Human references vs. model error (see <a href="#">paper</a> )

The length distribution of the subsets with a Llama tokenizer is shown below.

Subset	Chosen Mean Tokens	Reject Mean Tokens	Chosen Max Tokens	Reject Max Tokens	Chosen Min Tokens	Reject Min Tokens	Chosen Mean Unique Tokens	Reject Mean Unique Tokens	Chosen Max Unique Tokens	Reject Max Unique Tokens	Chosen Min Unique Tokens	Reject Min Unique Tokens
alpaca-eval-easy	591.26	167.33	1332	1043	40	15	252.91	83.44	630	290	33	12
alpaca-eval-hard	411.684	136.926	1112	711	57	12	172.537	70.9684	359	297	45	8
alpaca-eval-length	510.589	596.895	1604	2242	55	52	192.442	188.547	434	664	30	38
donotanswer	169.61	320.5	745	735	20	20	103.743	156.941	358	337	18	13
hep-cpp	261.262	259.488	833	835	53	57	99.8537	99.372	201	201	37	40
hep-go	266.22	264.598	732	720	55	57	99.622	99.189	201	201	36	37
hep-java	263.14	260.939	748	733	55	54	102.311	101.927	207	206	39	41
hep-js	251.165	249.695	771	774	53	52	93.2744	92.9268	192	192	37	40
hep-python	211.988	211.146	624	612	53	49	85.6463	85.3049	190	190	36	35
hep-rust	221.256	219.049	988	993	46	49	95.1402	94.8354	192	192	36	36
l1mbar-adver-GPTInst	170.109	377.359	636	959	15	15	92.9457	179.37	287	471	12	13
l1mbar-adver-GPTOut	96.4255	101	393	476	18	20	60.0426	55.0426	241	228	13	14
l1mbar-adver-manual	159.804	264.37	607	737	23	33	91.9565	140.13	273	385	18	24
l1mbar-adver-neighbor	70.2239	172.507	603	865	9	13	43.3134	90.9328	250	324	8	9
l1mbar-natural	139.42	129.82	907	900	17	18	74.99	70.07	354	352	14	14
math-prm	279.313	488.841	1608	1165	35	77	83.6264	124.582	237	257	23	46
mt-bench-easy	391.821	481.929	778	1126	155	31	169.071	121.321	288	434	74	19
mt-bench-hard	287.784	301.649	573	1176	68	62	133.622	121.676	261	309	50	48
mt-bench-med	351.375	466.025	655	1297	145	52	159.9	140.325	285	495	82	41
refusals-dangerous	208.4	458.61	380	804	87	103	128.53	211	200	365	71	55
refusals-offensive	139.82	298.63	278	1117	75	26	95.98	134.02	170	477	60	19
xstest-should-refuse	129.227	217.019	402	549	18	15	80.5519	116.149	194	245	16	13
xstest-should-respond	188.708	107.356	515	465	20	16	103.788	67.328	231	202	15	16

**Filtering Summary** The RewardBench dataset is manually filtered from 5123 source prompts to manually verify the chosen-rejected ranking of prompts.

- The categories of AlpacaEval and MT Bench are manually filtered for every prompt.
- LLMBar, DoNotAnswer, HEP, and Math PRM all contained structured metadata for automatic filtering.
- XSTest is a hybrid of manual confirmation with metadata from the project.
- Refusals are automatically generated as a refusal or response (where refusal is preferred) with manual confirmation.

Substantial filtering details are available in the appendix of the paper. If there are any bugs in the data, please reach out! No PII was collected during the manual annotation.

**License information** Licensing an aggregated dataset is a complex task. We release the RewardBench dataset under [ODC-BY](#) requiring the user to follow the licenses of the subsequent parts. Licensing LLM datasets is an evolving topic. The licenses primarily apply to the prompts and the completions generated by models are often unlicensed. The details for the datasets used in this work vary in the level of the detail on licenses and method of applying them.

Dataset	Variants	Data License
AlpacaEval	{Easy, Length, Hard}	CC By NC 4.0
MT Bench	{Easy, Medium, Hard}	Apache 2.0
LLMBar	{Natural, Neighbor, GPTInst, GPTOut, Manual}	MIT License
Do Not Answer		CC BY NC SA 4.0
XSTest	{Should Respond, Should Refuse}	CC By 4.0
HEP	{CPP, Go, Javascript, Rust, Python, Rust}	MIT License
PRM		MIT License
Math		

Within this dataset are prompts created by AI2 (the refusals data, released as MIT for now, see official release soon) with completions from API and open models. More details will come on this soon.

#### A.4.2 Development

**Requirements** Building the dataset requires datasets. Maintaining the script and notebooks requires notebook.

```
pip install datasets notebook nbconvert
```

Convert with:

```
jupyter nbconvert --to script [YOUR_NOTEBOOK].ipynb
```

With no changes to the ipynb, the dataset can be re-built and pushed with the following (PLEASE BE CAREFUL):

```
python build_dataset.py
```

**Git LFS notes** If your uploads fail with:

```
Git LFS upload failed: 14% (1/7), 4.2 MB | 0 B/s
(missing) data/train-00000-of-00001.parquet (425c88744455a9b0e7248cdd81fe4716085aae22849798f653
hint: Your push was rejected due to missing or corrupt local objects.
hint: You can disable this check with: 'git config lfs.allowincompletepush true'
```

First fetch all lfs objects:

```
git lfs fetch --all origin main
```

**Filtering script (basic)** To filter data, run the following script:

```
python scripts/filter.py subset-name 0
```

with a subset from the dataset and a start index.

---

### A.4.3 Citation

```
@misc{RewardBench,  
  title={RewardBench: Evaluating Reward Models for Language Modeling},  
  author={Lambert, Nathan and Pyatkin, Valentina and Morrison, Jacob and Miranda, LJ and Lin, H},  
  year={2024},  
  howpublished={\url{https://huggingface.co/spaces/allenai/reward-bench}}  
}
```