Dynaboard: An Evaluation-As-A-Service Platform for Holistic Next-Generation Benchmarking

Zhiyi Ma[†*] **Kawin Ethayarajh**[‡*] **Tristan Thrush**[†*] **Somya Jain**[†]

Ledell Wu[†] **Robin Jia**[†] **Christopher Potts**[‡] **Adina Williams**[†] **Douwe Kiela**[†]
[†] Facebook AI; [‡] Stanford University
dynabench@fb.com

Abstract

We introduce Dynaboard, an evaluation-as-a-service framework for hosting benchmarks and conducting holistic model comparison, integrated with the Dynabench platform. Our platform evaluates NLP models directly instead of relying on self-reported metrics or predictions on a single dataset. Under this paradigm, models are submitted to be evaluated in the cloud, circumventing the issues of reproducibility, accessibility, and backwards compatibility that often hinder benchmarking in NLP. This allows users to interact with uploaded models in real time to assess their quality, and permits the collection of additional metrics such as memory use, throughput, and robustness, which – despite their importance to practitioners – have traditionally been absent from leaderboards. On each task, models are ranked according to the Dynascore, a novel utility-based aggregation of these statistics, which users can customize to better reflect their preferences, placing more/less weight on a particular axis of evaluation or dataset. As state-of-the-art NLP models push the limits of traditional benchmarks, Dynaboard offers a standardized solution for a more diverse and comprehensive evaluation of model quality.

1 Introduction

Benchmarks have been critical to driving progress in AI: they provide a standard by which models are measured, they support direct comparisons of different proposals, and they provide clear-cut goals for the research community. This has led to an outpouring of new benchmarks designed not only to evaluate models on new tasks, but also to address weaknesses in existing models [49, 55, 34], and expose artifacts in existing benchmarks [42, 19, 25, 33, 22, 37]. These efforts are helping to provide us with a more realistic picture of how much progress the field has made.

To date, the metrics by which we assess system performance have received much less systematic attention. Even as the benchmarks have changed, the community has continued to rely heavily on accuracy as the sole primary metric. This gives rise to a "leaderboard culture" in which accuracy is the only thing that matters, even in contexts in which other pressures – e.g., compactness, fairness, efficiency – are clearly important [60, 6, 16, 63, 68, 38, 2, 51, 17]. Recently, Ethayarajh and Jurafsky [17] provided a microeconomic framing of the problem: leaderboard viewers are *consumers* of models, and each viewer has their own set of preferences: some care only about accuracy, others value compactness in addition to accuracy, and so on. Static leaderboards that only rank by model performance thus have a scoring function that is misaligned with the preferences of most users. Moreover, merely focusing on a single metric, such as accuracy, limits the scope of possible issues

[*]Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

even to consider. In other words, focusing on one standard metric, like accuracy, lets researchers off the hook too much on other pressing issues.

Reproducibility, accessibility, and backwards compatibility are important issues as well. There is a reliance on self-reported results with no trust guarantees [75], and many state-of-the-art claims are famously difficult to reproduce [52]. In some cases, the state-of-the-art model cannot be accessed without sufficient computational resources and/or technical expertise, and older models are often incomparable to newer ones because they have not been evaluated on the same data.

To begin to address these issues, we propose **Dynaboard**. Dynaboard's evaluation-as-a-service backend allows models to be submitted directly to be evaluated in the cloud, circumventing issues of reproducibility, accessibility, and backwards compatibility. Older models can be compared to newer ones because any model can be evaluated on demand. Dynaboard supports reasoning about many different metrics – not just standard accuracy-style metrics, but also memory use and throughput (i.e., inference speed) on identical hardware, robustness, fairness, and so forth. Building on [17], we borrow from microeconomic theory to aggregate these metrics – along with performance – into a *Dynascore* that is used to rank models. The frontend of Dynaboard is a dynamic leaderboard that allows users to customize the Dynascore by placing more/less weight on a particular metric or dataset. As the user modifies the weights that govern the Dynascore, models are re-ranked in real time. Users can also directly interact with a model via the platform, receiving real-time model predictions that help them understand a model's capabilities and limitations, and allowing the community to find challenging examples where the current state of the art still falls short. Together, this allows users to develop and directly express the complex and interrelated values they have for their models.

Although other platforms – CodaLab, DAWNBench [10], and ExplainaBoard [39], to name a few – address some subset of the problems we describe above, Dynaboard is the first to address all of them in a single end-to-end system. It requires minimal overhead for model creators wishing to submit their model for evaluation, but offers maximal flexibility for users wishing to make fine-grained comparisons between models. As state-of-the-art NLP models push the limits of traditional benchmarks, Dynaboard offers a standardized solution for creating the next generation of benchmarks in a manner that allows for the diverse and comprehensive evaluation of model quality.

2 Objectives

Dynaboard aims to address the following issues in our current model evaluation paradigm:

Reproducibility The reliance on self-reported results with no trust guarantees [75] makes many state-of-the-art claims difficult to reproduce [52]. Even the choice of random seed can lead to substantially different results [15], and improvements on the state-of-the-art are often not statistically significant [8]. Implementational differences in evaluation metrics can also lead to different scores [54]. Given that the test data for static benchmarks is often publicly available, reported results may also be the outcome of overfitting hyperparameters [4].

Accessibility Whichever model happens to hold the "state-of-the-art" title should be accessible by as many researchers as possible. Democratizing model evaluation is essential to understanding our current weaknesses, for making progress in the long tails of the data distribution, and for collecting new adversarial datasets where the state-of-the-art fails. Just open sourcing the model weights is insufficient, because even when given trained models, inference-time compute resources may not be readily available in many parts of the world.

Backwards Compatibility Dynamic human-in-the-loop data collection is a great alternative to static benchmarks, but as new evaluation sets are introduced in later rounds, old models become incomparable to new ones, simply because they haven't been evaluated on the same data. It should be possible to evaluate *a model* on demand, rather than the model's *predictions* at a single point in time.

Forwards Compatibility Automated metrics such as BLEU and ROUGE have many known flaws, and creating better automated metrics (e.g., BLEURT [64], BERTScore [77]) is an active area of research. However, the current paradigm does not allow for old models to be evaluated on new automated metrics that come later, since it relies on reporting from the model creator. If our field comes up with new and better metrics, we should be able to immediately gauge overall model quality.

Prediction Costs Most leaderboards treat the cost of making predictions as zero, which does not hold in practice; memory use, latency, throughput, and lack of robustness – among many other factors – have real-world implications in deployment [60, 17]. A highly accurate model may be useless to an embedded systems engineer if it is untenably large, for example. For users to be able to make informed decisions about the risks and rewards of using a particular model, they need to be given as much information as possible.

Utility Estimation There is no single correct way to rank models: every leaderboard viewer has a different set of preferences, as manifested in their utility function [17]. Some users only care about model performance, but others care about memory use, throughput, fairness, and more. A leaderboard should not be overly prescriptive when ranking models – it should provide a default ranking, but ultimately give users the freedom to customize the scoring/ranking function to better approximate their utility function.

The evaluation-as-a-service backend of Dynaboard addresses the first four of these. By having models submitted directly to the platform, they can be accessed and evaluated on demand on any dataset (see Section 4). The frontend of Dynaboard is a dynamic leaderboard that addresses the last two issues: prediction costs and utility estimation. Although the leaderboard has a default ranking, the scoring function is constructed in a principled way, borrowing from economic theory to estimate the rates at which users are willing to trade-off prediction costs for performance. More importantly, users are given the freedom to manipulate the scoring function to better approximate their preferences.

3 Related Work

Evaluation platforms Although other platforms have addressed some subset of the issues in Section 2, they have not addressed all of them in a single end-to-end solution. DAWNBench [10] reports some prediction costs, such as the time needed to reach a particular accuracy. However, it still relies on self-reported results, meaning that reproducibility, accessibility, and forwards/backwards compatibility are still issues. CodaLab[2] and EvalAI[3] make it significantly easier to access models and reproduce stated claims, but automatically evaluating any model on any new dataset or metric is not straightforward, making backwards/forwards compatibility hard to scale. It would also be difficult to collect prediction costs on standard hardware. Explainaboard [39] and Robustness Gym [23] allow for a more nuanced comparison of models, but through the lens of which slices of data a model performs well on and what types of errors it makes. Models are not directly uploaded, however, so prediction costs are not collected and forwards compatibility cannot be automated. We see these approaches as complementary to Dynaboard, helping provide a richer view of model performance.

Multi-metric evaluation There have been some attempts to rank models by metrics other than accuracy. Although not explicitly framed in terms of utility, Mieno et al. studied the trade-off users were willing to make between accuracy and latency in speech translation [44]. Some challenge tasks in recent years have required the reporting of prediction costs or have instituted limits on the maximum costs that can be incurred, as in the EfficientQA challenge [27, 45]. However, such practices are still few and far between. Our hope is that over time Dynaboard can help lower the overhead required to create multi-metric benchmarks and help them proliferate and diversify, such that every sub-community can have its own canonical benchmarks, like GEM [21] for natural language generation, BIG-Bench[4] for language model probing, or GLUE [73] and SuperGLUE [72] for NLU.

4 Backend: Evaluation-as-a-Service

The evaluation-as-a-service pipeline uses a standardized application programming interface (API) that is similar to other model evaluation platforms like EvalAI and Codalab that allow creators to upload models. Researchers submit their models for evaluation in our *model evaluation cloud*, in which all models are scored in exactly the same way, on the exact same data, within the same (virtual) hardware constraints. Testing models on different datasets, from standard well-known test sets to

[2] https://codalab.org

[3] https://eval.ai

[4] https://github.com/google/BIG-bench

stress tests [47, 11] and check lists [59] can be cumbersome – we offer evaluation-as-a-service (EaaS) for NLP models. We ask model developers to specify model cards [46] to report on and properly document their contributions. In this paradigm, it is no longer possible for people to cherrypick results: the same model instance is evaluated on all evaluation sets.

We take a *multi-metric* approach, going beyond just accuracy, where models are evaluated on multiple axes of evaluation. The top models can be employed in-the-loop for dynamic adversarial evaluation [34], and all uploaded models can be made accessible online so that anyone can "talk to" these models (in a corresponding frontend UI) to see how they perform. Models are evaluated on combinations of adversarial and non-adversarial datasets. All statistics are made available to the user, who can rank models by specifying which datasets and which metrics should factor into the ranking and in what proportion (see Section 5 for details).

Importantly, we do not claim to have built or discovered the one right approach for model evaluation. The goal of Dynaboard – and the Dynabench data collection platform it is built on – is making *dynamic* as many of the components employed by the current status quo as possible: i) rather than having a single metric, we can have many metrics, with the ability to add more as new automated metrics are conceived; ii) rather than having a fixed set of evaluation datasets, we can have many and add more over time; iii) rather than evaluating models only on static datasets, they can be evaluated dynamically in the loop. For this approach to succeed, the ability to re-evaluate older models is essential: when new datasets are introduced, new metrics come out, and as new community preferences emerge, we need to be able to see where we stand with our current state-of-the-art.

4.1 Metrics

All models are evaluated along multiple axes using a variety of metrics. The ideal model is not only accurate, but fast, memory-efficient, fair and robust. Many of these properties are associated with trade-offs: a binary classifier that always outputs 1 is very fast and uses little memory, but is very inaccurate. While we might incorporate additional metrics in the future – one of the advantages of our platform is that we are able to do so – the currently reported set of metrics is as follows:

Performance The standard evaluation metric in machine learning is some form of performance, whether it is accuracy, F1, AUROC, BLEU, or something else. The exact performance metric, in our case, is task dependent. One task can have multiple performance metrics, but only one metric is used as the canonical performance metric when ranking models. When metrics are aggregated to rank models, our default weighting places half the weight on the canonical performance metric and splits the remaining half among the others (see Section 5 for details).

Throughput The amount of inference-time compute required by a given model is defined as the number of examples it can process per second on its instance in our evaluation cloud. For fair comparison, models are deployed on the exact same architecture. Measuring the computational efficiency of a model is important for two main reasons. First, a highly accurate model that takes a very long time to label a single example has limited use in most real-world scenarios. Second, as a field, we need to focus more on *Green AI* and explicitly account for a model's energy/carbon footprint in our model ranking [63, 68, 28].

Memory The amount of memory that a model requires is given in gigabytes of memory usage, averaged over the duration that the model is running, with measurements taken over a fixed interval. Note that unlike with the other metrics, the goal is to minimize memory use rather than maximizing it. For this reason, before aggregating all the metrics into a single score for ranking models, we transform "memory used" into "memory saved" by subtracting it from the maximum possible (virtual) memory that can be used (see Section 5 for details).

Fairness. There is no single, best way to measure model fairness [35, 50, 3]. Whether to include any explicit fairness metric was a difficult choice – we acknowledge that we might inadvertently facilitate false or spurious fairness claims, fairness-value hacking, or give people a false sense of fairness. The research community has a long way to go in terms of developing well-defined concrete fairness measurements. Ultimately, however, we came to the conclusion that fairness is such an important axis of evaluation that we would rather have an imperfect metric than no metric at all: in our view, a multi-metric evaluation framework simply *must* include fairness as a primary axis.

Table 1: Scoring and evaluation-as-a-service datasets used for each of the four Dynabench tasks. Unless otherwise indicated, scoring sets are the respective test sets.

Task	Scoring	Evaluation-as-a-Service
NLI	SNLI [5], MNLI [74] matched and mismatched, ANLI rounds 1-3 [49]	Respective dev sets; HANS [43], NLI stress tests [47], Winogender [62] recast as NLI [53]
QA	SQuAD dev [57], AdversarialQA [1] (Dynabench QA round 1)	AdversarialQA dev, the 12 dev sets from MRQA shared tasks [18]
Sentiment	SST3 [66], DynaSent [55]	Respective dev sets; Amazon Reviews [78] test and dev (10k subsample), Yelp Reviews [78] test and dev (10k subsample)
Hate speech	Learning From The Worst [71]	Respective dev sets; HateCheck [61]

We compute the fairness metric based on demographic parity using a black box evaluation protocol, without assuming any access to the model itself beyond getting its predictions. Thus the question becomes: how do we construct or manipulate input data such that we can assess some aspects of a model's fairness by inspecting its outputs? Following the precedent set by datasets such as Winogender [62] and WinoBias [79], we perform perturbations of original datasets in two ways: (i) by substituting noun phrases that unambiguously encode a gender identity with those that have the same part-of-speech but refer to another identity (e.g., replacing "he" with "they" following [14, 13]) and (ii) by substituting names with others that are statistically predictive [70] of another race or ethnicity (e.g., replacing "James" with "Jamal"). This way, a model is considered more 'fair' if its predictions do not change after perturbation, and considered less fair if they do. More details are in Appendix A. As the community invents better and better black box fairness evaluation metrics, which we hope they will, we can re-evaluate older models on new datasets and new metrics, due to our backwards-compatible evaluation paradigm. Note that while we use fairness as a metric, we can and do also report results on fairness datasets [62] as part of our evaluation-as-a-service platform.

Robustness. This is a question of how robust a model is to a particular set of (mostly demographic) axes, and there are many other factors that can be involved in a model's robustness. In order to capture this aspect of model quality, we also conduct black box robustness evaluation in a similar manner to fairness. That is, following past work on NLP robustness [30, 31], we perturb examples and measure whether a model's prediction changes. Specifically, we use the recently released TextFlint[5] evaluation toolkit [24] to do so (details are provided in Appendix B). Initially, we focus mostly on typographical errors and local paraphrases – e.g., a "baaaad restuarant" is not a good restaurant – but just like for fairness, we fully expect this metric to evolve and improve over time.

4.2 Tasks

Currently, there are leaderboards for four tasks: Natural Language Inference (NLI), Question Answering (QA), Sentiment Analysis and Hate Speech. We evaluate on a combination of adversarially collected datasets for these four tasks, as well as datasets from other sources. Each task has its own set of task owners who control the default setting, i.e., which datasets are included in the corresponding benchmark and in the evaluation suite, how the datasets are weighted, and how the metrics are weighted relative to each other for that particular task. Note that the leaderboard rank is only determined based on a subset of the full set of evaluation-as-a-service datasets (see Table 1).

5 Frontend: Dynamic Leaderboard

The frontend of Dynaboard aggregates the various metrics into a single score for ranking models. Users can manipulate the parameters of this scoring function to better reflect their preferences.

A static leaderboard's ranking cannot approximate the preferences of the leaderboard viewer, because it cannot consider (e.g. computational) costs. Ethayarajh and Jurafsky [17] frame this shortcoming in economic terms: a leaderboard viewer is a *consumer* of models and the benefit they get from a model

[5]https://www.textflint.com

is its *utility* to them – each individual has their own utility function. The scoring function implicit in ranking by accuracy or F1 is misaligned with the utility function of most users; for example, an IoT engineer looking to find the fastest and smallest model getting 80% accuracy would be poorly served by a static leaderboard.

In reporting many prediction costs, Dynaboard addresses the first of these problems. But how do we combine these disparate metrics into a single score that can be used to rank models? Additionally, how can we allow Dynaboard users to align the scoring function with their own utility function? We propose a method for calculating "exchange rates" between metrics that can be used to standardize units across metrics, after which a weighted average is taken to get the Dynascore. The user can adjust the default weights – e.g., our IoT engineer can place more emphasis on memory and throughput – to approximate their utility function; as they do so, the models will be dynamically re-ranked.

5.1 Background

Each model has k properties that informs its utility – throughput, memory, accuracy, etc. – which can be thought of as a separate *good*. By definition, a good is something one always wants more of, meaning that "memory saved" is a good but "memory used" is not, for example. When a metric is not naturally a good, we transform it into a good by subtracting it from a budget cap or the maximum of that metric across all models. For example, subtracting memory used from a maximum available memory of 16 GB gives us "memory saved", which we want to maximize. This standardization allows us to use the same method when calculating the trade-off between any two metrics.

A model is a point in this space of goods, and a user's utility function maps each point to an amount of utils. An *indifference curve* is a set of points that provide the same utility and can be thought of as a level curve of the utility function [41]. Indifference curves are monotonically negatively-sloped: if one model is strictly better on all dimensions than another, then the two cannot lie on the same negatively sloped line, because the former will have strictly greater utility. For example, for a more accurate model to be on the same curve as a less accurate one, the former may have to use up more memory. For a given indifference curve, the rate at which this trade-off is made – i.e., the negative of its slope at a point – is called the *marginal rate of substitution* (MRS) [41].

5.2 Converting between Metrics

To calculate the default Dynascore, we first estimate the rate at which users are willing to trade-off each metric for a one-point gain in performance (i.e., MRS with respect to performance) and use that to convert all metrics into units of performance. Once converted, a weighted average is taken to get the final score. We choose this approach for two reasons. First, by drawing from the microeconomics literature, it is a principled approach to user personalization that builds on the utility-based critique of static leaderboards [17]. Second, typical normalization methods (e.g., z-scores) do not work well when a metric's values are highly skewed, as they often are in practice. For example, as seen in Table 2, taking a weighted average of z-scores would lead to T5 ranking worse than the majority baseline on many tasks; this happens because T5's high memory use is as much an outlier as the baseline's poor accuracy. Even BERT, a more lightweight model, ranks worse than the majority baseline on NLI and QA when ranking by the average z-score, which is highly unintuitive.

Given that each user has their own unique utility function – and thus unique indifference curves – each has a different rate at which they are willing to make a trade-off between a metric `M` and the performance metric `perf`. This is fine; the default "exchange rates" between the metrics should merely be a starting point that can be adjusted by the user as they see fit (see Section 5.3). To calculate the default exchange rates between some metric `M` and `perf`, we consider the simplest possible case, where each indifference curve only lies in two dimensions – i.e., the utility changes only with respect to `M` and `perf` – and all models lie on the same indifference curve. This assumption is needed because we do not have direct access to any k-dimensional utility function; our data is sparse.

We then calculate the *average marginal rate of substitution* (AMRS) of this indifference curve, which tells us the rate at which model creators, as a group, are trading off `M` for a one-point increase in `perf` while keeping utility constant. For example, if AMRS of "memory saved" with respect to accuracy were 0.5 GiB, then each GiB of memory saved would on average be worth 2 points of accuracy. By dividing `M` by $\mathbf{AMRS}(\texttt{M}, \texttt{perf})$, we can convert it to units of performance.

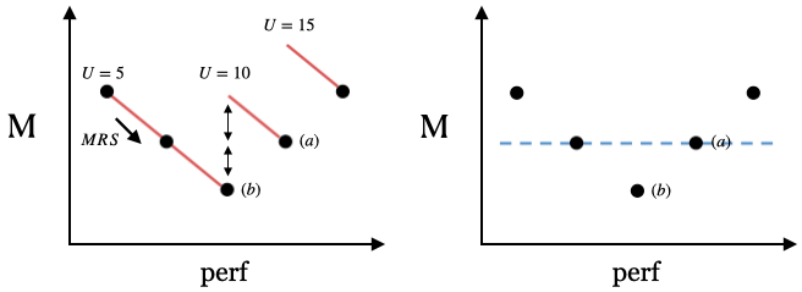


Figure 1: Left: The marginal rate of substitution (MRS) is the negative of the slope of an indifference curve (shown in red) with fixed utility U, each of which shows the trade-off being made between metric $\mathtt{M}$ and performance $\mathtt{perf}$. Since model (a) and (b) cannot possibly lie on the same curve – because the former is better in both respects – we assume that when all else (including utility) is held constant, the increase in performance from (b) to (a) should come at the *expense* of the increase in $\mathtt{M}$, giving us an estimate of where the higher indifference curve lies. Right: Taking the line-of-best-fit to estimate this trade-off would not work when some models are strictly better than others.

Note that this means that the Dynascore is inherently dynamic and incorporates the AMRS *at a given point in time*, and so is subject to change as more models are added.

Assumptions What if there were no trade-off between a metric and performance? For example, performance might rise with robustness up to a point and come at the cost of robustness past that. If the ith most accurate model x_i had both better performance and were more robust than the next most accurate model x_{i+1}, then they could not possibly lie on the same indifference curve, since having more of both goods is axiomatically preferable to having less of each. To tackle this problem, we make two assumptions: (i) if $\mathtt{M}(x_i) > \mathtt{M}(x_{i+1})$ and $\mathtt{perf}(x_i) > \mathtt{perf}(x_{i+1})$, then models $\{x_j | j > i\}$ lie on a lower indifference curve than x_i, though not all necessarily lie on the same one; (ii) there exists a model $\langle \mathtt{perf}(x_{i+1}), \mathtt{M}(x_i) + (\mathtt{M}(x_i) - \mathtt{M}(x_{i+1}))\rangle$ on the same indifference curve as x_i (i.e., x_i and x_{i+1} would be on the same curve if the latter obtained $\mathtt{M}(x_i) - \mathtt{M}(x_{i+1})$ more on metric $\mathtt{M}$). In other words, all else (including utility) held constant, we assume that the increase in performance from x_{i+1} to x_i should come at the *expense* of the change in $\mathtt{M}$. This is just an estimate, but it provides a good starting point from which users can choose their own AMRS (Section 5.3).

These assumptions, visualized in Figure 1, allow every model to be represented in the AMRS calculation. Under these assumptions about where the higher indifference curve lies, calculating the MRS is equivalent to taking the absolute value of the slope between points (rather than just taking the negative of the slope)[6]:

$$\begin{aligned} \textbf{MRS}(\mathtt{M}, \mathtt{perf}) &= \left\{ \left| \frac{\mathtt{M}(x_i) - \mathtt{M}(x_{i+1})}{\mathtt{perf}(x_i) - \mathtt{perf}(x_{i+1})} \right| \mid 1 \leq i < n \right\} \\ \textbf{AMRS}(\mathtt{M}, \mathtt{perf}) &= \overline{\textbf{MRS}} \end{aligned} \tag{1}$$

5.3 Weights and Customization

After converting all the metrics into units of performance, we take a weighted average of the converted values to get the Dynascore for a model. For a model x_i with normalized weights $w_{\mathtt{M}} \in \mathbb{Z}$:

$$\mathtt{Dynascore}(x_i) \triangleq \sum_{\mathtt{M}} w_{\mathtt{M}} \cdot \frac{\mathtt{M}(x_i)}{\textbf{AMRS}(\mathtt{M}, \mathtt{perf})} \tag{2}$$

The default normalized weights are $w_{\mathtt{perf}} = 0.5$ and $w_{\mathtt{M} \neq \mathtt{perf}} = 0.5/(m-1)$. In other words, performance makes up half the Dynascore while the remaining half is split equally among the other metrics. This is because, in theory, gains in all the other metrics may come at the expense

[6]Since the MRS is undefined when $\mathtt{perf}(x_i) - \mathtt{perf}(x_{i+1})$ is zero, we ignore increases in performance that fall below a predetermined threshold epsilon, or are ϵ-small. We currently set ϵ to 1×10^{-4} for all tasks.

Table 2: Non-dynamic leaderboards for the Dynabench tasks, sorted by the Dynascore. Performance (Perf.) is measured by F1. Note that ranking by the average z-score (with the same metric and dataset weighting as the Dynascore) leads to a far more unintuitive ranking than with Dynascore – e.g., T5 often ranks worse than the majority baseline because its memory use is such an outlier. Also note that although "memory used" is reported, before incorporating it into the Dynascore and z-score, we subtract it from the maximum possible 16GB to get "memory saved" to be consistent and express it as a utility (i.e., having something we want to maximize).

Task	Model	Perf.	Throughput	Memory	Fairness	Robustness	Dynascore	Avg z-score
NLI	DeBERTa	69.54	7.41	5.71	91.97	75.70	38.83	0.24
	RoBERTa	69.07	9.23	4.82	90.94	74.82	38.61	0.24
	ALBERT	67.29	9.60	2.18	89.94	74.12	37.72	0.26
	T5	67.16	7.10	10.62	91.89	73.47	37.53	-0.07
	BERT	64.82	9.39	4.13	92.11	66.38	36.36	0.06
	Majority Baseline	32.41	77.33	1.15	100.00	100.00	22.78	0.10
	FastText	31.29	73.94	2.20	83.23	69.14	21.13	-0.83
QA	DeBERTa	76.25	4.47	6.97	88.33	90.06	45.92	0.48
	ELECTRA-large	76.07	2.37	25.30	93.13	91.64	45.79	0.33
	RoBERTa	69.67	6.88	6.17	88.32	86.10	42.54	0.27
	ALBERT	68.63	6.85	2.54	87.44	80.90	41.74	0.16
	BERT	57.14	6.70	5.55	91.45	80.81	36.07	-0.02
	BiDAF	53.48	10.71	3.60	80.79	77.03	33.96	-0.44
	Unrestricted T5	28.80	4.51	10.69	92.32	88.41	22.18	-0.52
	Return Context	5.99	89.80	1.10	95.97	91.61	15.47	-0.27
Sentiment	DeBERTa	76.07	7.50	4.80	94.08	79.21	71.31	0.34
	RoBERTa	73.74	8.95	4.14	93.87	77.81	70.11	0.28
	T5	73.20	7.12	9.06	93.44	77.99	69.32	0.00
	ALBERT	70.61	10.24	2.04	93.34	78.44	68.73	0.28
	BERT	68.71	8.83	6.04	93.49	72.75	66.81	-0.07
	Majority Baseline	35.93	35.14	1.07	100.00	100.00	57.94	-0.27
	FastText	53.32	32.54	1.69	78.52	65.82	57.39	-0.57
Hate Speech	DeBERTa	81.34	6.20	5.40	83.58	81.94	48.31	0.23
	RoBERTa	80.26	6.61	3.67	85.02	79.09	47.77	0.26
	ALBERT	76.84	8.01	2.25	84.50	79.98	46.18	0.23
	BERT	76.58	7.26	3.22	86.45	77.35	45.97	0.15
	T5	76.59	5.48	10.47	86.71	78.52	45.80	-0.19
	Majority Baseline	54.69	16.48	1.09	100.00	100.00	37.05	0.24
	FastText	49.70	15.34	2.61	80.09	71.12	32.46	-0.93

of performance, so giving each metric a weight of $1/m$ would favor degenerate models due to their efficiency and close-to-zero memory. To avoid this, the weighting was split evenly between performance and the (hypothetical) costs of performance.

A Dynaboard user can customize the Dynascore to better approximate their own utility function by adjusting the un-normalized weights. In doing so, they are effectively expressing the average rate at which *they* would be willing to trade off metric $\mathtt{M}$ against performance. That is, using the user-specified normalized weights $z_{\mathtt{M}}$ is equivalent to using the default normalized weights with an effective exchange rate of $(w_{\mathtt{M}}/z_{\mathtt{M}})\mathbf{AMRS}(\mathtt{M}, \mathtt{perf})$. If a Dynaboard user cares more about a metric $\mathtt{M}$ than in our default setting (i.e., $w_{\mathtt{M}} < z_{\mathtt{M}}$), then the adjusted AMRS will decline to reflect the fact that they are less willing to sacrifice $\mathtt{M}$ for a marginal increase in performance. A possible line of future work is studying NLP practitioners' preferences over time and across different niches (e.g., industry vs. academia). Asking such questions has only been made possible by adopting this utility-based framework for Dynascore.

5.4 Limitations & Future Work

In general, the AMRS estimates are better when our assumptions in 5.2 need not apply – i.e., when the models can all lie on the same convex indifference curve. However, we stress that the estimates are only a starting point and are meant to be manipulated by the user by adjusting the weights.

Identical Models Since every metric is converted into units of performance, if *all* models perform exactly the same, the AMRS cannot be calculated. The reason we use performance as the base metric is because every task is guaranteed to have *some* performance metric, while the other metrics may differ substantially across tasks. However, it is possible to pick a different base metric in principle: we could calculate the AMRS of $\mathtt{M}$ w.r.t. memory saved, for example, and convert everything into

GiB. Similarly, the converted value for `M` will be undefined if all models obtain exactly the same value for `M` – this implies that model creators are not willing to sacrifice `M` at all for an improvement in performance and therefore that the metric is infinitely more valuable relative to others. Both these issues are unlikely to occur in practice, however.

Threshold Sensitivity We choose to ignore increases in performance less than $\epsilon = 1 \times 10^{-4}$ so that the MRS is never undefined, which we consider reasonable since almost all existing leaderboards only report to 1 or 2 decimal points, so an improvement of 1×10^{-4} would be indistinguishable to the user. One could reasonably argue that this threshold needs to be higher however, since many NLP datasets are too small for a 0.01 or even 0.1 increase to be statistically significant [8]. Changing ϵ can change the Dynascore and the model rankings. However, to avoid potentially confusing the user with adaptive thresholds based on statistical power tests, we chose to fix ϵ at a conservative value.

Human Validation Not all models and tasks lend themselves to be evaluated automatically; some need humans in-the-loop, either by definition or over time as our models improve. In the future, we hope to incorporate the validated model error rate (vMER) [34] – the number of human-validated model errors divided by the total number of examples – as a metric on the leaderboard, such that any benchmark can use vMER as a base performance metric.

5.5 Results

We take models that currently represent the state-of-the-art on our tasks and put them through our evaluation pipeline. Specifically, we finetune and evaluate BERT [12], RoBERTa [40], ALBERT [36], T5 [56] and DeBERTa [26] on all tasks, with the addition of ELECTRA [9] for QA. This set of models roughly encompasses the top 5 models on GLUE [73] and SuperGLUE [72]. As baselines, we add FastText [32] for sentiment and hate speech and BiDAF [65] for QA. We also compare to the majority baseline for the classification tasks (using the training set majority) and for QA a baseline that simply returns the entire context. These baselines are obviously efficient and robust to perturbations, but have low performance. See Appendix C for further details. We compute the Dynascore for each model using the default weights for the metrics and giving equal weight to all datasets. Recall that the Dynascore is dynamic and incorporates the AMRS at a given point in time, and so is subject to change as more models are added.

The results are reported in Table 2. We find that the SuperGLUE ranking is roughly preserved, with Transformer models clearly outperform their predecessors and the baselines. Even when we factor in the additional axes of evaluation, DeBERTa still performs best. In some cases, the gap is actually larger: in terms of accuracy on SuperGLUE, DeBERTa outperforms T5 by 1.1% , but on the sentiment task it's 3.1%, while being more efficient, as reflected in the Dynascore. However, there are some differences with the SuperGLUE rankings, most notably with T5, which ranks much lower here because of its relatively low throughput and high memory use. It is also worth noting that FastText does consistently worse than the majority baseline, even when it is more accurate, since the gain in accuracy is not enough to offset its sensitivity to fairness and robustness perturbations.

The somewhat old-fashioned medium of a paper does not do justice to a dynamic concept like a leaderboard with specifiable utility function, and linking to the actual implementation would break anonymity. We refer the reviewer to the supplementary material for examples.

6 Conclusion

We introduced an end-to-end solution for hosting next-generation benchmarks enabling a more holistic and comprehensive evaluation of model quality. Our evaluation-as-a-service platform addresses several important shortcomings in the current status quo, from reproducibility to accessibility to backwards/forwards compatibility. Dynaboard also enables the collection of prediction costs and factors these costs into the overall ranking of models, which users can customize to better align with their own preferences. With NLP models playing an increasingly important role in our daily lives, deciding which model is better than another one is becoming a crucial problem, both for driving further progress and ensuring that we deploy our systems responsibly. We hope that Dynaboard can allow benchmarks to proliferate and diversify, paving the way for the next wave of advances in NLP.

Acknowledgments and Disclosure of Funding

We thank our collaborators in the Dynabench team, especially Max Bartolo, Yixin Nie and Bertie Vidgen, for helpful comments and suggestions. We're grateful to Amanpreet Singh and Sujit Verma for useful feedback and engineering suggestions. We thank Eric Smith and April Bailey for helping with name lists for the fairness perturbation. We also thank Devi Parikh, Alicia Parrish, Pedro Rodriguez, Dan Jurafsky, Ethan Perez, Kyunghyun Cho and Alex Wang for comments on an earlier draft, and Gargi Ghosh for her support throughout. We thank the TextFlint team for their help on the toolkit. Kawin Ethayarajh was supported by an NSERC PGS-D.

Broader Impact

Green AI & Ethical AI A key barrier to the adoption of Green AI [63] has been the incentive structure in NLP leaderboard culture, which places emphasis on higher performing (i.e., more accurate) models to the detriment of models that are more lightweight and efficient [17]. Although there are exceptions to this – such as the EfficientQA challenge task [45] – the dominant paradigm still largely ignores these factors. Changing incentive structures is important because they ultimately shape what kind of problems the NLP community works on and what kind of models they build. By drastically lowering the overhead needed to host multi-metric benchmarks, Dynaboard will hopefully allow benchmarks to keep fulfilling their crucial role in driving research progress, while incentivizing the creation of models that are "greener" and more fair, among other qualities.

Fairness Our perturbation-based black box fairness evaluation method is an initial attempt at measuring fairness for NLP tasks/models, and we fully expect this to evolve over time as advances are made in the field. Firstly, our method assumes that a fair system will be one that treats all genders and races/ethnicities equally. We take equal treatment (treating all groups equally) to be a first step towards a longer term goal of equitable treatment (treating all groups fairly in accordance with their needs or circumstances). The equal treatment perspective implies that fairness perturbations of the input should not affect the classification label. In most cases, this assumption is valid; however, this is not always true, and even when it is, does not necessarily apply equally to all members of the group referenced in the example (e.g., not everyone who identifies as a woman can or chooses to give birth). This makes our measurements noisier. We do not currently use our perturbation method to filter out any examples from our data, because this may unintentionally lead to the over- or under-representation of certain topics and voices.

Secondly, whenever one chooses a set of demographics, they are immediately codifying particular categories and deciding which groups to include or leave out. We have chosen for now only a set of size two—gender identity and race/ethnicity. We chose these largely because they have clear and measurable effects in language (i.e., with respect to pronoun morphology, noun phrases, and names). Within these two broad groups, we have chosen to include only a subset of the possible perturbations: man-woman for gender, and Asian-Pacific Islander-Black-Hispanic-white for race/ethnicity). For the latter, this decision was driven by data availability and could reflect a US-centric bias present in the existing datasets. For the former, we intended to additionally include a gender-neutral perturbation, which could swap names that were statistically more likely to refer to men or women to those that were gender-balanced, and similarly to perturb "sister" and "brother" to "sibling", "her" and "him" to "them", and "female" and "male" to the empty string. We feel this direction is of clear importance, since it can make it possible to uncover examples that are unfair to nonbinary people. Unfortunately, there were various idiosyncrasies both in our language ("standard" American English), and in the existing datasets that made the quality of the examples decrease too drastically for us to conclude that a gender-neutral perturbation would yield useful signal. For example, for the "(fe)male" swap with the empty string, we observed that 50% of the uses of these terms in MNLI were nouns; permuting them to the empty string would result in ungrammatical examples. Furthermore, in addition to case collapses between possessive forms of the pronouns (i.e., "their(s)", singular-*they* has an additional complication in that it controls plural verbal agreement, meaning that heuristic perturbations often introduce additional ungrammatical noise above and beyond the case collapses in morphological form). Due to these and other complications, we reluctantly set aside gender-neutral perturbations for future work. All this being said, there is ample room to improve upon our current fairness perturbations, not only by extending the scope of included groups, but also by exploring options for perturbation that are more flexible.

Thirdly, our fairness perturbation method might have different consequences for different tasks. For Sentiment Analysis, QA, and NLI, our spot checks suggest that data crucial for the classification is affected only rarely (see Appendix A for more discussion). However, for hate speech, the situation is a more nuanced. In data collection [71], abuse against men, straight people, white people, etc. was not considered (either as hateful or non-hateful). By performing heuristic perturbations of gender identity and race/ethnicity words, we post hoc include these sorts of examples. Since they have never seen such examples at training, this may overestimate model sensitivity to perturbation.

Finally, we would like to re-emphasize that there are many ways to measure and conceive of fairness of NLP models. We have chosen to adopt a perspective where the gender and race/ethnicity of entities in examples should not affect classification decisions. We think this is valid for the tasks hosted on Dynabench so far. This being said, models that perform well on our fairness perturbations should not be taken to necessarily be "fair" models, although we do feel it is safe to conclude that any models that perform poorly on these perturbations have room for improvement.

Dynascore As discussed in Section 5, the Dynascore of a model is not fixed. Rather, it is dynamic on many levels: (i) as new models are added, the rates at which metrics are converted to units of performance may change; (ii) as the weights change, the model rankings will change. Therefore the default score is not a fixed and canonical measure of a model, and model creators should avoid reporting the Dynascore of their model out-of-context: a Dynascore only has meaning in the context of all the other models on the same leaderboard. However, users may not appreciate these caveats, and they may report the default Dynascore of their model in a research paper, much in the way they would report accuracy or F1. To discourage such behavior, the Dynaboard UI will provide clear warnings to the user, informing them of the limitations of Dynascore and the fairness and robustness metrics. If any Dynascore is reported, we recommend that users explicitly report minimally the dataset, the metric weights, and the timestamps.

References

[1] Max Bartolo, Alastair Roberts, Johannes Welbl, Sebastian Riedel, and Pontus Stenetorp. Beat the AI: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678, 2020.

[2] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

[3] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, 2020.

[4] Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. In *International Conference on Machine Learning*, pages 1006–1014. PMLR, 2015.

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

[6] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[7] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[8] Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, 2020.

[9] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *ICLR*, 2020.

[10] Cody Coleman, Deepak Narayanan, Daniel Kang, Tian Zhao, Jian Zhang, Luigi Nardi, Peter Bailis, Kunle Olukotun, Chris Ré, and Matei Zaharia. Dawnbench: An end-to-end deep learning benchmark and competition. In *NIPS ML Systems Workshop*, 2017.

[11] Ishita Dasgupta, Demi Guo, Andreas Stuhlmüller, Samuel J. Gershman, and Noah D. Goodman. Evaluating compositionality in sentence embeddings. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, 2018.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, 2019.

[13] Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188, 2020.

[14] Emily Dinan, Angela Fan, Ledell Wu, Jason Weston, Douwe Kiela, and Adina Williams. Multi-dimensional gender bias classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 314–331, 2020.

[15] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*, 2020.

[16] Ravit Dotan and Smitha Milli. Value-laden disciplinary shifts in machine learning. *arXiv preprint arXiv:1912.01172*, 2019.

[17] Kawin Ethayarajh and Dan Jurafsky. Utility is in the eye of the user: A critique of NLP leaderboards. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4846–4853, 2020.

[18] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*, 2019.

[19] Markus Freitag, David Grangier, and Isaac Caswell. BLEU might be guilty but references are not innocent. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 61–71, 2020.

[20] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[21] Sebastian Gehrmann, Tosin P. Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh D. Dhole, Wanyu Du, Esin Durmus, Ondrej Dusek, Chris Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Rubungo Andre Niyongabo, Salomey Osei, Ankur P. Parikh, Laura Perez-Beltrachini, Niranjan Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Sellam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabezudo, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. The GEM benchmark: Natural language generation, its evaluation and metrics. *arXiv preprint arXiv:1609.05807*, 2021.

[22] Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th*

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1161–1166, 2019.

[23] Karan Goel, Nazneen Rajani, Jesse Vig, Samson Tan, Jason Wu, Stephan Zheng, Caiming Xiong, Mohit Bansal, and Christopher Ré. Robustness Gym: Unifying the NLP evaluation landscape. *arXiv preprint arXiv:2101.04840*, 2021.

[24] Tao Gui, Xiao Wang, Qi Zhang, Qin Liu, Yicheng Zou, Xin Zhou, Rui Zheng, Chong Zhang, Qinzhuo Wu, Jiacheng Ye, et al. Textflint: Unified multilingual robustness evaluation toolkit for natural language processing. *arXiv preprint arXiv:2103.11441*, 2021.

[25] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, 2018.

[26] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[27] Kenneth Heafield, Hiroaki Hayashi, Yusuke Oda, Ioannis Konstas, Andrew Finch, Graham Neubig, Xian Li, and Alexandra Birch. Findings of the fourth workshop on neural generation and translation. In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 1–9, 2020.

[28] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research*, 21(248):1–43, 2020.

[29] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[30] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, 2017.

[31] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4120–4133, 2019.

[32] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, 2017.

[33] Divyansh Kaushik and Zachary C Lipton. How much reading does reading comprehension require? A critical investigation of popular benchmarks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5010–5015, 2018.

[34] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking Benchmarking in NLP. In *Proceedings of NAACL*, 2021.

[35] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016.

[36] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

[37] Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. Question and answer test-train overlap in open-domain question answering datasets. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1000–1008, 2021.

[38] Tal Linzen. How can we accelerate progress towards human-like linguistic generalization? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5210–5217, Online, July 2020. Association for Computational Linguistics.

[39] Pengfei Liu, Jinlan Fu, Yang Xiao, Weizhe Yuan, Shuaicheng Chang, Junqi Dai, Yixin Liu, Zihuiwen Ye, and Graham Neubig. EXPLAINABOARD: An Explainable Leaderboard for NLP. *arXiv preprint arXiv:2104.06387*, 2021.

[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arxiv*, abs/1907.11692, 2019.

[41] N Gregory Mankiw. *Principles of economics*. Cengage Learning, 2020.

[42] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. Tangled up in BLEU: Reevaluating the evaluation of automatic machine translation evaluation metrics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4984–4997, 2020.

[43] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, 2019.

[44] Takashi Mieno, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. Speed or accuracy? a study in evaluation of simultaneous speech translation. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[45] Sewon Min, Jordan Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, et al. NeurIPS 2020 EfficientQA Competition: Systems, Analyses and Lessons Learned. *arXiv preprint arXiv:2101.00133*, 2021.

[46] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[47] Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2340–2353, 2018.

[48] Leonard S. Newman, Mingxuan Tan, Tracy L. Caldwell, Kimberley J. Duff, and E. Samuel Winer. Name norms: A guide to casting your next experiment. *Personality and Social Psychology Bulletin*, 44(10):1435–1448, 2018.

[49] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[50] Alexandra Olteanu, Kartik Talamadupula, and Kush R. Varshney. The limits of abstract evaluation metrics: The case of hate speech detection. In *Proceedings of the 2017 ACM on Web Science Conference*, page 405–406, 2017.

[51] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M Bender, Emily Denton, and Alex Hanna. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *arXiv preprint arXiv:2012.05345*, 2020.

[52] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *arXiv preprint arXiv:2003.12206*, 2020.

[53] Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, 2018.

[54] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, 2018.

[55] Christopher Potts, Zhengxuan Wu, Atticus Geiger, and Douwe Kiela. DynaSent: A dynamic benchmark for sentiment analysis. *arXiv preprint arXiv:2012.15349*, 2020.

[56] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[57] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[58] Adithya Renduchintala and Adina Williams. Investigating failures of automatic translation in the case of unambiguous gender. *arXiv preprint arXiv:2104.07838*, 2021.

[59] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, 2020.

[60] Anna Rogers. How the transformers broke NLP leaderboards. `https://hackingsemantics.xyz/2019/leaderboards/`, 2019. Accessed: 2020-05-20.

[61] P. Röttger, B. Vidgen, Dong Nguyen, Z. Waseem, H. Margetts, and J. Pierrehumbert. Hatecheck: Functional tests for hate speech detection models. *ArXiv*, abs/2012.15606, 2020.

[62] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 8–14, 2018.

[63] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green AI. *arXiv preprint arXiv:1907.10597*, 2019.

[64] Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, 2020.

[65] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations*, 2017.

[66] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[67] Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. Astraea: Grammar-based fairness testing. *arXiv preprint arXiv:2010.02542*, 2020.

[68] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, 2019.

[69] Konstantinos Tzioumis. Data for: Demographic aspects of first names, 2018.

[70] Konstantinos Tzioumis. Demographic aspects of first names. *Scientific data*, 5(1):1–9, 2018.

[71] Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. Learning from the worst: Dynamically generated datasets to improve online hate detection. *ArXiv*, abs/2012.15761, 2020.

[72] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, 2019.

[73] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[74] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[75] Craig Willis and Victoria Stodden. Trust but verify: How to leverage policies, workflows, and infrastructure to ensure computational reproducibility in publication. *Harvard Data Science Review*, 2(4), 12 2020.

[76] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[77] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2019.

[78] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 2015.

[79] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, 2018.

[80] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 15–20, 2018.

[81] Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. Learning gender-neutral word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4847–4853, 2018.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] (Users upload only a single model instance)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Fairness metric

Dynabench comprises four dynamic tasks with multiple rounds of datasets that will grow over time. Given that here we have to be able to evaluate a wide variety of models, both in the loop and outside of it, we employ a black box post hoc approach, i.e., one that can be applied post-data collection to existing data, on any uploaded model, without requiring anything other than its predictions. One straightforward way to measure fairness then, is to apply clearly delimited, heuristic perturbations to existing evaluation datasets, and measure whether performance drops. Such an approach is similar to recent works that use grammars to heuristically generate pairs of examples varying in gender [58] and/or race [67] in that they utilize predefined lists of words. However, because we also want to ensure minimal consequences on our classification labels, we adopted an approach that is more targeted than grammars and also preserves the original input data distribution: we replace each word in the input data that has a clear signal about race/ethnicity and/or gender identity with a similar word referring to another group, rerun inference, and measure how many labels flipped (i.e., the difference in microaverage accuracy).

For race/ethnicity, we seeded first names across four demographic groups from a public dataset of 4,250 first names from US mortgage lending applications [69, 70]. These names cover 85.6 percent of the U.S. population, are based on the 1990 Census information on first name frequencies [70], and are licensed under a very permissive license (Creative Commons Attribution 4.0 International License). Names were associated with mutually exclusive demographic groups recognized by the 2000 and 2010 US Government Census—we selected the four groups with the most names: Asian/Pacific Islander, Black, Hispanic, white. Note: there is nothing inherently "racial" about particular names—for example, each demographic group had at least a few people named "Anna" or "Benjamin" [70]—although there are statistical trends. We selected all names for which a plurality of people of that name identified with one of the races or ethnicities, then took the 200 most frequent. We then also augmented that list with additional names popular in the literature [7, 48]. To construct our permuted test dataset, whenever we encountered a name in the input data from one of our lists, we randomly selected a name from a different race/ethnicity list and substituted it. Whenever we encountered a name in the input data which was not present in our list of names, we left it unperturbed.

For gender identity, we investigate two kinds of perturbations: names and noun phrases. For names, we affiliated the names from our race/ethnicity list with statistically likely genders, based on the U.S. Social Security Association's Lists of Baby Names (1980–2019), and performed perturbations as we did for race/ethnicity.[7] For noun phrases (i.e., pronouns and nouns), we adopted a slightly more structured approach: we still replaced words based on pair-based word lists, but we didn't do so randomly, as that could result in ungrammatical sentences (e.g., one can't replace a pronoun, like "her" with a noun like "dad" and expect no effect on the classification label). Words in the paired list that either exclusively referred to women (e.g., *her*, *sister*) or to men (e.g., *his*, *brother*) were selected by taking the union of existing popular word lists [80, 81] that had been recently extended [13, 14].[8]

Given that our perturbations are heuristic, some noise is to be expected. For example, for the names perturbations, content relevant for the classification can be affected when the name is part of a phrase referring to a known named entity. Consider "I've always enjoyed eating at *Red Robin*" being perturbed to "I've always enjoyed eating at *Red Kayla*". To mitigate this issue, we first ran an off-the-shelf named entity recognition system, and did not perturb any examples for which the system found a familiar named entity.[9] We observe that there are very few noisy examples (less than 5) resulting from NER errors. Finally, there is one irreducible type of noise arising from our heuristic approach—perturbing gender-explicit information occasionally results in unusual examples and may have consequences for the classification label: For NLI, the following hypothesis "Mothers should nurse at night" became "Fathers should nurse at night" in the context of "Failing to nurse at night can lead to painful engorgement or even breast infection". Based on spot checks performed by the authors, we conclude that noise resulting from explicit gender information is also rare (only three out of 122 spot-checked sentences). Although our approach yields enough signal to evaluate whether model performance depends on race/ethnicity and gender identity, future work exploring more flexible and adaptable approaches is encouraged.

[7] `https://www.ssa.gov/oact/babynames/`

[8] For a discussion of non-binary gender, see the Broader Impact Statement.

[9] We use the NER pipeline from spaCy [29], which was trained on OntoNotes 5.

B Robustness metric

For the robustness evaluation, we use a post hoc black box approach similar to the fairness evaluation. We use TextFlint [24], an open source library for measuring model robustness that covers a wide range of text transformations. We apply a family of universal transformations from TextFlint, namely `Contraction`, `Keyboard`, `Ocr`, `Punctuation`, `SpellingError`, `Typos` and `WordCase`, as we focus on typographical errors for our robustness perturbations.

The robustness metric is computed as the percentage of unchanged predictions before and after perturbation. The assumption is that a robust model should not change its predictions upon such perturbations in input. That is to say, a model is deemed more robust if it has a higher robustness metric (i.e., a lower difference between original and perturbed examples).

C Model Details

We selected and trained a diverse set of models to demonstrate the value of the platform, as well as to provide a sense of the current state of the art across our multiple metrics. We did not tune hyperparameters, instead using default training hyperparameters.

Training Data The NLI models are trained on ANLI [49] combined with MNLI [74], where ANLI is upsampled by a ratio of 2 to balance the data. The QA models are trained on SQuAD [57] combined with Adversarial QA [1]. The hate speech models are trained on rounds 0 through 4 of the Learning from the Worst dataset [71], with an upsampling ratio for each of 1, 5, 100, 1, and 1, respectively. The sentiment models are trained on of Dynasent [55], where round 2 was upsampled by a ratio of 3. Learning from the Worst [71], provides the upsample ratios used for hate speech, and we also performed early stopping for the hate speech transformers with the round 4 dev set. The rest of the upsampling was done to give the Dynabench data more influence in the training routine, as the leaderboard test sets on our evaluation platform are mostly comprised of Dynabench data.

Transformers All of the transformer models are the base versions provided in the HuggingFace transformers library [76], except ELECTRA which is the large discriminator version. ELECTRA-large was used to test the capacity of our framework to handle larger transformer models.

We used the default training hyperparameters from HuggingFace's transformers training scripts.

The T5 models were trained with early stopping on SQuAD + Adversarial QA for QA, and the last round of the Dynabench data for the other tasks. For 2-way sequence classification, the model was trained to predict the first token of one of the strings "0", "1". For 3-way sequence classification, it was trained to predict the first token of one of the strings "0", "1", "2". There is no universal default way to handle which token to output from the T5 for sequence classification, as every sequence classification task is different. We used the training hyperparameters from the HuggingFace community T5 for text classification example.[10] An exception is that we trained for 3 epochs for text classification to match the BERT-style model default (the default for T5 is 2).

FastText All of the FastText models were the text classification version trained with an initial learning rate of 1, 25 epochs, 2-grams, bucket size of 200000, 50 dimensions, and hierarchical SoftMax. This is the default, for large datasets, that is listed on FastText's website.[11]

BiDAF BiDAF was trained with the default settings from the AllenNLP [20] BiDAF trainer.[12]

Majority Baseline The majority label was determined from the corresponding train set. For example, the MNLI train set has frequency-matched labels, and the ANLI train set has entailed, neutral, contradictory splits of 52,111 / 68,789 / 41,965. So the majority baseline for NLI was to always return neutral.

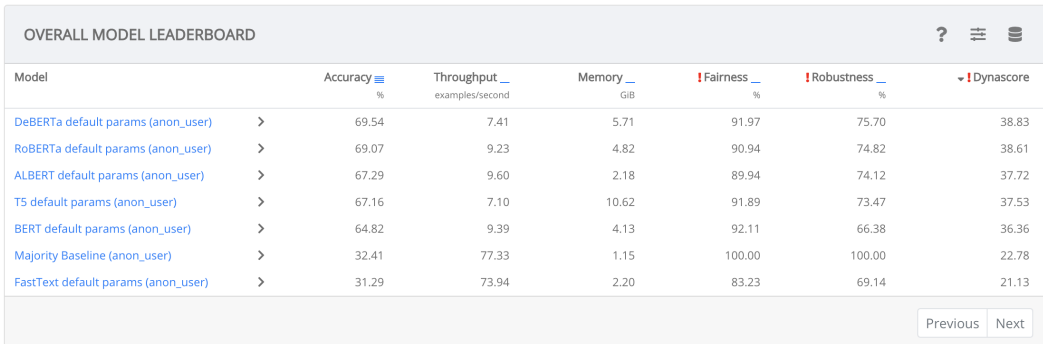
[10] See `https://huggingface.co/transformers/community.html` and `https://github.com/patil-suraj/exploring-T5`.

[11] `https://fasttext.cc/docs/en/supervised-tutorial.html`

[12] `https://github.com/allenai/allennlp`

D Screenshots

Please also see the screen recording provided in the supplementary material to get a better sense of how the interface works.

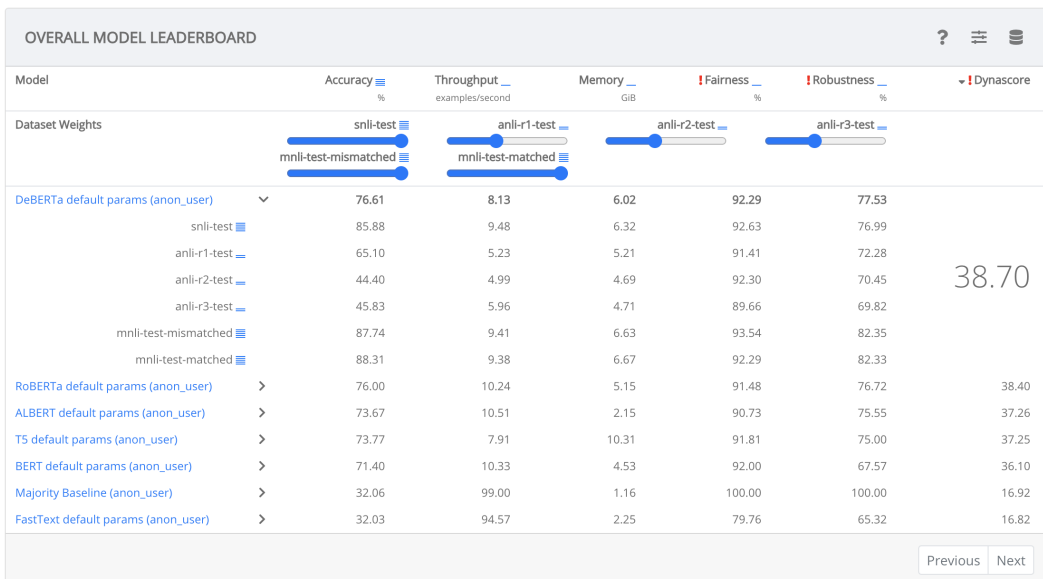


OVERALL MODEL LEADERBOARD

Model	Accuracy %	Throughput examples/second	Memory GiB	Fairness %	Robustness %	Dynascore
DeBERTa default params (anon_user)	69.54	7.41	5.71	91.97	75.70	38.83
RoBERTa default params (anon_user)	69.07	9.23	4.82	90.94	74.82	38.61
ALBERT default params (anon_user)	67.29	9.60	2.18	89.94	74.12	37.72
T5 default params (anon_user)	67.16	7.10	10.62	91.89	73.47	37.53
BERT default params (anon_user)	64.82	9.39	4.13	92.11	66.38	36.36
Majority Baseline (anon_user)	32.41	77.33	1.15	100.00	100.00	22.78
FastText default params (anon_user)	31.29	73.94	2.20	83.23	69.14	21.13

Previous Next

Figure 2: NLI Dynaboard example.



OVERALL MODEL LEADERBOARD

Model	Accuracy %	Throughput examples/second	Memory GiB	Fairness %	Robustness %	Dynascore
Dataset Weights	snli-test, mnli-test-mismatched	anli-r1-test, mnli-test-matched	anli-r2-test		anli-r3-test	
DeBERTa default params (anon_user)	76.61	8.13	6.02	92.29	77.53	
snli-test	85.88	9.48	6.32	92.63	76.99	
anli-r1-test	65.10	5.23	5.21	91.41	72.28	
anli-r2-test	44.40	4.99	4.69	92.30	70.45	38.70
anli-r3-test	45.83	5.96	4.71	89.66	69.82	
mnli-test-mismatched	87.74	9.41	6.63	93.54	82.35	
mnli-test-matched	88.31	9.38	6.67	92.29	82.33	
RoBERTa default params (anon_user)	76.00	10.24	5.15	91.48	76.72	38.40
ALBERT default params (anon_user)	73.67	10.51	2.15	90.73	75.55	37.26
T5 default params (anon_user)	73.77	7.91	10.31	91.81	75.00	37.25
BERT default params (anon_user)	71.40	10.33	4.53	92.00	67.57	36.10
Majority Baseline (anon_user)	32.06	99.00	1.16	100.00	100.00	16.92
FastText default params (anon_user)	32.03	94.57	2.25	79.76	65.32	16.82

Previous Next

Figure 3: NLI Dynaboard example, with dataset weight sliders and finer-grained metrics displayed.

OVERALL MODEL LEADERBOARD

Metric Weights	Accuracy	Throughput	Memory	Fairness	Robustness	Dynascore
DeBERTa default params (anon_user)	69.54	7.41	5.71	91.97	75.70	38.83
RoBERTa default params (anon_user)	69.07	9.23	4.82	90.94	74.82	38.61
ALBERT default params (anon_user)	67.29	9.60	2.18	89.94	74.12	37.72
T5 default params (anon_user)	67.16	7.10	10.62	91.89	73.47	37.53
BERT default params (anon_user)	64.82	9.39	4.13	92.11	66.38	36.36
Majority Baseline (anon_user)	32.41	77.33	1.15	100.00	100.00	22.78
FastText default params (anon_user)	31.29	73.94	2.20	83.23	69.14	21.13

Previous Next

Figure 4: NLI Dynaboard example, with metric weight sliders in default configuration.

OVERALL MODEL LEADERBOARD

Metric Weights	Accuracy	Throughput	Memory	Fairness	Robustness	Dynascore
RoBERTa default params (anon_user)	76.66	10.43	5.24	91.56	77.03	15.76
DeBERTa default params (anon_user)	77.16	8.27	6.06	92.34	77.78	15.75
ALBERT default params (anon_user)	74.17	10.69	2.15	90.79	75.75	15.44
T5 default params (anon_user)	74.49	8.06	10.25	91.85	75.28	15.13
BERT default params (anon_user)	72.23	10.52	4.57	92.06	67.81	14.98
Majority Baseline (anon_user)	32.00	102.94	1.16	100.00	100.00	11.84
FastText default params (anon_user)	32.22	98.32	2.25	79.02	64.64	11.17

Previous Next

Figure 5: NLI Dynaboard example, with both dataset weights and metric weight sliders modified.

Leaderboard Datasets

snli-test	85.88
mnli-test-mismatched	87.74
mnli-test-matched	88.31
anli-r1-test	65.10
anli-r2-test	44.40
anli-r3-test	45.83

Non-Leaderboard Datasets

superglue-winogender	59.55
mnli-dev-mismatched	88.27
mnli-dev-matched	89.13
snli-dev	85.49
hans	73.68
nli-stress-test	77.56
anli-r1-dev	63.90
anli-r2-dev	46.20
anli-r3-dev	45.42

Figure 6: DeBERTa on NLI example, part of the model page, showing non-leaderboard evaluation-as-a-service dataset results as well.