Evaluate & Evaluation on the Hub:
Better Best Practices for Data and Model Measurements

Leandro von Werra, Lewis Tunstall, Abhishek Thakur, Alexandra Sasha Luccioni,
Tristan Thrush, Aleksandra Piktus, Felix Marty, Nazneen Rajani, Victor Mustar, Helen Ngo, Omar Sanseviero, Mario Šaško,
Albert Villanova, Quentin Lhoest, Julien Chaumond, Margaret Mitchell, Alexander M. Rush, Thomas Wolf, Douwe Kiela
Hugging Face, Inc.
{leandro,lewis,abhishek,sasha.luccioni,douwe}@huggingface.co

Abstract

Evaluation is a key part of machine learning (ML), yet there is a lack of support and tooling to enable its informed and systematic practice. We introduce Evaluate and Evaluation on the Hub—a set of tools to facilitate the evaluation of models and datasets in ML. Evaluate is a library to support best practices for measurements, metrics, and comparisons of data and models. Its goal is to support reproducibility of evaluation, centralize and document the evaluation process, and broaden evaluation to cover more facets of model performance. It includes over 50 efficient canonical implementations for a variety of domains and scenarios, interactive documentation, and the ability to easily share implementations and outcomes. The library is available at https://github.com/huggingface/evaluate. In addition, we introduce Evaluation on the Hub, a platform that enables the large-scale evaluation of over 75,000 models and 11,000 datasets on the Hugging Face Hub, for free, at the click of a button. Evaluation on the Hub is available at https://huggingface.co/autoevaluate.

Demo screencast: youtube.be/6rU177zRj8Q

1 Introduction

Evaluation is a crucial cornerstone of machine learning—not only can it help us gauge whether and how much progress we are making as a field, it can also help determine which model is most suitable for deployment in a given use case. However, while the progress made in terms of hardware and algorithms might look incredible to a ML practitioner from several decades ago, the way we evaluate models has changed very little. In fact, there is an emerging consensus that in order to meaningfully track progress in our field, we need to address serious issues in the way in which we evaluate ML systems (Kiela et al., 2021; Bowman and Dahl, 2021; Raji et al., 2021; Hutchinson et al., 2022).

Figure 1: Average number of evaluation datasets and metrics per paper, based on 10 random samples per year from EMNLP proceedings over the past two decades. More recent papers use more datasets and metrics, while fewer of them report statistical significance test results.

In order to have a clearer idea regarding the way model evaluation has evolved in our field, we have carried out our own analysis on a random sample of EMNLP papers from the past two decades, and present our results in Figure 1. It can be observed that the number of evaluation datasets and metrics per paper has increased over time, suggesting that model evaluation is becoming increasingly complex and heterogeneous. However, auxiliary techniques such as testing for significance, measuring statistical power, and using appropriate sampling methods have become less common, making results harder to judge when comparing new results to previous work. We believe that while datasets are now more easily accessible thanks to shared repositories (Lhoest et al., 2021), model evaluation is still unnecessarily cumbersome, with a fragmented ecosystem and a lack of consensus around evaluation approaches and best practices.

The goal of this work is to address three practical challenges in model evaluation for ML: reproducibility, centralization, and coverage.

Reproducibility: ML systems are extremely sensitive to small (and often undocumented) choices.
such as random seeds and hyperparameters (Pineau et al., 2021). Model performance is often not compared with proper statistical testing that takes this variance into account, making many self-reported comparisons unreliable. Our goal is to standardize this process and thereby improve the reproduction of ML evaluations.

Centralization: Historically, ML metrics have been poorly documented, exacerbating an already insufficient community-wide understanding of their usage and shortcomings (Post, 2018). As metrics and datasets change, the onus is on the community to keep results up-to-date, causing unnecessary replication of work (Ma et al., 2021) and the proliferation of outdated artifacts (Luccioni et al., 2022).

Coverage: ML as a field still focuses heavily on accuracy-based metrics. While important, this focus glosses over other critical facets such as efficiency (Min et al., 2021), bias and fairness (Qian et al., 2022), robustness (Goel et al., 2021), and how these factor into choosing a model (Ethayarajh and Jurafsky, 2020; Ma et al., 2021).

We introduce the open source Evaluate library and the Evaluation on the Hub platform to address many of these problems. We believe that better evaluation can happen, if we—as a community—establish better best practices and remove hurdles.

2 Related work

Open-Source Tools for Evaluation There is a long history of open source projects aiming to capture various measurements, metrics and statistical testing methods for ML. Torchmetrics (DetlefSEN et al., 2022) implements a large number of model evaluation metrics for PyTorch (Paszke et al., 2019), which is similar to evaluation metrics found in Keras (Chollet et al., 2015) for TensorFlow. Libraries like Scikit-learn (Pedregosa et al., 2011), SciPy (Virtanen et al., 2020), Statsmodels (Seabold and Perktold, 2010), NLTK (Bird et al., 2009), TrecTools (Palotti et al., 2019), RL Reliability Metrics (Chan et al., 2020), NetworkX (Hagberg et al., 2008), Scikit-image (Van der Walt et al., 2014), GEM (Gehrmann et al., 2021), TorchFidelity (Obukhov et al., 2020) also support many evaluation measures across many domains. As integrating metrics into specific frameworks can be difficult, there are also many libraries dedicated to individual evaluations for example rouge_score, BartScore (Yuan et al., 2021), or SacreBLEU (Post, 2018). The fragmentation of the ecosystem leads to various problems, such as a wide range of incompatible conventions and APIs, or misreporting due to differing implementations and results.

In Evaluate, we provide one single interface backed by a centralized Hub. Metrics can easily be shared, are version controlled, have a standardized interface, and allow for multimodal inputs.

Evaluation as a Service The idea of Evaluation as a Service (Ma et al., 2021; Kiela et al., 2021), whereby models are submitted for another party to be centrally evaluated, has recently gained traction as a more reproducible way to conduct model evaluation. Central evaluation also facilitates holding challenges and competitions around datasets (Yadav et al., 2019; Pavao et al., 2022; Akhbardeh et al., 2021) as opposed to simply evaluating self-reported model results or comparing model scores with benchmark suites (Bajaj et al., 2016; Coleman et al., 2017; Wang et al., 2018, 2019; Kardas et al., 2020; Reddi et al., 2020; Liu et al., 2021; Goel et al., 2021; Dror et al., 2019). The advantages of conducting evaluation centrally are multiple, including better reproducibility, forward/backward compatibility, and the ability to measure models along multiple axes of evaluation (e.g. efficiency and fairness, in addition to accuracy), which can help contribute towards a more systematic approach to evaluation.

Issues with Evaluation Several studies of ML research and practice have been carried out in recent years on different aspects pertaining to ML evaluation, and together they paint a bleak picture of evaluation in our field. For instance, a 2019 large-scale replication study of 255 ML papers found that only 63% of the results they reported could be systematically replicated (Raff, 2019). A complementary survey of 3,800 papers from Papers with Code has shown that a large majority of metrics used do not adequately reflect models’ performance and that they largely do not correlate with human judgement (Blagec et al., 2021). Finally, a recent study of 770 papers in machine translation from the last decade found that while 108 new metrics have been proposed for the task, 99.8% of papers continue to use BLEU score for reporting results (Marie et al., 2021), despite the fact that the
original BLEU score (Papineni et al., 2002) has been shown to vary based on user-chosen parameters such as tokenization, which vary across languages (Post, 2018; Ananthakrishnan et al., 2007). These issues motivate the development of the tools presented in this work.

3 Library: Evaluate

The Evaluate library provides canonical implementations of a large set of evaluation modules. Modules are available to the community via a single, easy-to-use API. We provide extensive and detailed documentation cards for each, describing their correct usage, range of values and possible pitfalls, in a similar vein to model and dataset cards (Mitchell et al., 2019; Gebru et al., 2021). To facilitate extensibility, each evaluation model lives in a separate Git repository, and new modules can be easily contributed. The core library is released under the Apache 2.0 license and is available on GitHub, making it easy to adopt and deploy.

The library is designed to address the main challenges discussed in Section 1. Metrics are versioned and documented to support reproducibility within the framework. The core system is centralized to facilitate comparisons across models in a consistent manner supporting best practices, and data is stored in Git to allow backups and cloning. Finally, the tool is inherently designed for a multi-model, multi-evaluation paradigm supporting broad evaluation coverage by default.

3.1 Library Structure

Evaluate aims to support a range of model and dataset comparisons. It offers three distinct types of evaluation modules:

**Metrics:** Metrics to provide a score for model performance (e.g. accuracy or BLEU score). They play a central role for decisions around the use and deployment of models, allowing models to be compared and evaluated based on given benchmarks.

**Comparisons:** Comparisons are used to compare the predictions of two models (e.g. McNemar’s test). When comparing two models, these scores can help determine whether the difference in the models’ behavior is statistically significant.

**Measurements:** Measurements are used to investigate the characteristics of a dataset (e.g. fraction of duplicates, skew in label distribution). These statistics are a crucial step for gleaning more insights regarding training or evaluation datasets.

3.2 Library Tour

We demonstrate how Evaluate works with a quick tour of its features. In this section we focus on metrics, but the showcased methods work identically for the other types of evaluation modules.

**Core Library**  Any metric, measurement, or comparison can be loaded using its name.

```python
import evaluate
metric = evaluate.load("accuracy")
```

The name can refer to a local file path or the name of a repository on the Hugging Face Hub. Users can add predictions and/or references one at a time or pass all of them directly to compute().

```python
metric.add_batch(predictions = [1, 1], references = [1, 0])
metric.compute()
```

Note that the sequential method is particularly useful in a multi-worker setup, where each worker adds data and the compute operation happens at the end. Evaluate uses Apache Arrow as its backend, which means that adding data to the metric does not use any additional memory. The full set of data is only loaded when the metric is computed.

Several metrics can be bundled together and follow the same API as a single metric, returning all results at once.

```python
evaluate.combine(["accuracy", "f1"])
```

**Evaluator** Evaluate also offers a higher level API called the Evaluator. Evaluator enables anyone to quickly evaluate a model on a task. Evaluator encapsulates task-specific pre- and post-processing and streamlines data preparation, model inference and metric computation. This makes the evaluation of any (model, dataset, metric) triplet on a task seamless:

```python
task = evaluator("text-classification")
task.compute(model_or_pipeline=model, data=data, metric=metric)
```

Currently text, token, and image classification as well as question-answering are supported with more coming soon.
Evaluator employs pipelines from the Transformers library (or any other object with the same API) to carry out model inference. While evaluating downstream performance of the model, the Evaluator keeps track of the inference efficiency via metrics such as throughput and latency. This provides another dimension along which models can be compared, especially relevant in applied scenarios where inference times may be as crucial to a model’s success as its performance on the core metrics. The Evaluator also supports (optional) confidence interval computations via bootstrapping on any metric.

3.3 Documentation

Recent years have seen several proposals for standardized documentation of both models (Mitchell et al., 2019) and datasets (Gebru et al., 2021), arguing that this improves their accessibility as well as enabling a better understanding of their limitations and biases across different audiences. We have adopted this line of work within Evaluate—accompanying each evaluation module is a documentation card that describes the measurement, metric or comparison and how to use it. This card includes its intended use (i.e., whether it is specific to a task such as machine translation or a dataset such as SQuAD), its range, and code snippets that a user can copy within their application. These cards also contain a section on limitations and biases of the module, such as their applicability for certain languages (this is especially relevant for metrics such as BERTScore and COMET, which leverage pretrained models), the size of the models used to calculate them (e.g., GPT-2, the default model used for calculating MAUVE, is over 3 GB), and the fact that certain modules (e.g., perplexity) are not comparable across different datasets when built from different models or preprocessing steps.

Our goal with these documentation cards is two-fold. On the one hand, we hope that they will educate users regarding the scope and intention of different evaluation approaches, how they are calculated and how to interpret their values. On the other hand, we aim to improve best practices in terms of evaluation approaches. This can be as simple as measuring F1 score instead of relying simply on accuracy for imbalanced datasets, but also preferring a more reproducible and systematic metric such as SacreBLEU over a more variable one such as BLEU. We encourage the creators of new modules to write documentation cards to inform the community regarding the intended usages of their metric, measurement, or comparison; their possible limitations and biases; and to provide examples of best practices for using them.

3.4 Community Contributions

Since the code for metrics is stored in individual repositories on the Hugging Face Hub, anyone can add new metrics and load them with Evaluate without needing to wait for reviews or approval. Any piece of evaluation code can be easily pushed to the Hugging Face Hub, which allows for sharing the exact same implementation with direct collaborators and the broader research community. These community metrics complement the canonical modules and are stored under the user’s namespace. The Evaluate library also includes a command line interface (CLI) to make community contributions more accessible.

```
evaluate-cli create "My awesome metric"
```

This command creates a repository on the Hub, clones it, populates it with a template and pushes it to the Hub. The user only needs to implement the metric logic, write a README containing the metric card, and push their changes to the Hub using Git. We automatically provide live interaction widgets for each module, allowing users to develop a proper intuition for evaluation modules’ usage, along with access to their documentation. Furthermore, our community discussion feature allows members of the community to flag problematic evaluations or to ask for details regarding results, which model creators can then engage with.

4 Service: Evaluation on the Hub

The Evaluation on the Hub platform extends the Evaluate library to a free service model: anyone can evaluate any model on any dataset using any compatible metric, without requiring any code. This service utilizes models, datasets, and metrics standardized through the Hugging Face Hub. All evaluation results using this method are produced by the same pipeline with versioned implementations, and so are inherently reproducible. When a new model, dataset, or metric is produced, anyone can rerun the evaluation. As such, Evaluation on
the Hub facilitates large-scale evaluation of over 75,000 models and 11,000 datasets.

The service model further supports the goals of reproducibility and centralization. While the Evaluate library can ensure that the metrics used are consistent, it cannot ensure that the model was trained and evaluated using a reproducible set of hyperparameters and data. Incorporating Evaluate into a model hosting and training environment makes it possible to guarantee this consistency. Centralization also provides a further benefit of joining these metrics with model and data card documentation.

4.1 System architecture

The system architecture is shown in Figure 2. Upon submission, an evaluation job is triggered, which downloads the dataset and model(s) from the centralized Hub, computes metrics, and opens a pull request with the results.

Evaluation jobs are configured through a simple interface that specifies the task, dataset, metrics, and models to be evaluated. For each task, we compute a set of common metrics using the Evaluate library; users can also select additional metrics from the Hub to be included in the evaluation. For many datasets on the Hub, we provide evaluation metadata that defines a default configuration for users to launch evaluation jobs with a single click. Users can also add evaluation metadata to their own datasets to provide one-click evaluations to the community. The interface for triggering an evaluation is shown in Figure 3 (left).

We use AutoTrain, Hugging Face’s AutoML platform, to run evaluation jobs. The results from each evaluation are stored as metadata associated with model cards. The model predictions for each evaluation are also stored as dataset repositories on the Hub, enabling further analysis of, e.g., model errors.

4.2 Documenting Evaluation

The tool is permissioned so that model owners have the ability to select which evaluations they want to display with their model. This documentation is managed through a pull request system that allows owners to see evaluations that have been run. If a pull request is approved by the model owner, the results are added visibly to the model card as part of its documentation. However, all evaluation pull requests are public by default, so even if one is closed by model owners, members of the community can still see the scores.

Upon approval, the results become visible on an interactive Leaderboard associated with the underlying dataset. We aggregate all model evaluations (both verified and self-reported) through these leaderboards that allow users to filter results across task and dataset. Models are ranked so that users can find the best scoring model for task X on dataset Y. The interface for model leaderboards is shown in Figure 3 (left).

5 Use Cases

Evaluate and Evaluation on the Hub are already actively used by our community for a variety of tasks. There are many applications of these tools, and we highlight some of the most important use cases.
cases observed in practice.

Use case 1: Choosing the best model. If the task is known and the aim is to find an appropriate model, the Hub Leaderboard (which aggregates all the evaluation results for a dataset representative of that task) can act as a trusted source. In case a particularly interesting model is not yet on the leaderboard, its evaluation can easily be triggered, directly from the Hub, and its results will automatically appear on the leaderboard, allowing it to be compared to previous models.

Use case 2: Reproducibility of results. If a new dataset is created, it can be uploaded directly to the Hub to trigger evaluation coverage on many models without needing any code. Researchers can trust in the reproducibility and consistency of this evaluation of these models on this dataset. Similarly, open-source implementations for measurements, metrics and comparisons can easily be shared and plugged into the Evaluator to enable reproducibility on a range of other model facets. If a paper does not report results for a given model on a dataset of interest, it can be evaluated and verified.

Use case 3: Deciding on deployment. When deciding on which variant of a model to deploy to production, it is important to consider the broad performance of the model across multiple metrics. It may also be important to test on held-out test sets, and to measure the latency and throughput of a model. With the Evaluator, researchers can quickly evaluate on several datasets and also get the measured timing and latency information to make an informed decision.

Use case 4: Adding a new metric. When a new evaluation module (i.e., metric, measurement or comparison) is developed, it needs to be distributed for wider use. Historically, for use-cases like Kaggle competitions, metrics are shared as code snippets, requiring participants to copy the evaluation code, which can be error-prone and inconvenient. With Evaluate, anyone can create a new evaluate module—be it a metric, measurement, or comparison—alongside its documentation card with instructions. Anybody with the access rights can then quickly use the module with the standard loading mechanism.

6 Conclusion

Evaluate and Evaluation on the Hub aim to facilitate better evaluation of machine learning data and models by improving reproducibility, centralization, and coverage of evaluation tools. Evaluate is an open-source, community-driven library that standardizes evaluation. Evaluation on the Hub is a reproducible no-code alternative for evaluation across models, datasets, and metrics. We hope that this set of tools can help facilitate better best practices for model and data evaluation.

Ethical Issues and Limitations

There are multiple aspects of model evaluation that we have not (yet) addressed but that remain important in the broader landscape of our community and the way ML is used in real-world settings. For instance, we have currently focused on metrics and measurements that have been developed and tested for high-resource languages such as English, and only cover a handful of metrics that explicitly support multilinguality. Similarly, while we strove to cover as many metrics as possible, most of our coverage is for text-based metrics, and we have yet to

---

Example of a custom metric added by a community member: hf.co/co/spaces/jordyvl/ece
add as many metrics from other modalities, multimodal metrics, or to provide as large a selection for measurements and comparisons. Furthermore, while we have documented the computational and memory requirements of our evaluation approaches via documentation cards, several metrics require downloading large models such as GPT-2, which can be inaccessible for users with slower Internet speeds or insufficient memory. Finally, we are still working towards a greater reproducibility of evaluation results, for instance by adding identifiers that will indicate which version of a metric and dataset was used for evaluating a model (in the case of code changes, for instance), allowing users to easily replicate results if needed. We will continue improving our tools to address these limitations and provide support for more uses cases.

Acknowledgements

We thank the many contributors to the Hugging Face open source ecosystem. We thank Chunte Lee for designing the Evaluation on the Hub diagram in Figure 2.

References


Steven Bird, Ewan Klein, and Edward Loper. 2009. Natural language processing with Python: analyzing text with the natural language toolkit. ” O’Reilly Media, Inc.”. 


François Chollet et al. 2015. Keras. https://keras.io.


Adrien Pavao, Isabelle Guyon, Anne-Catherine Letourneil, Xavier Baró, Hugo Escalante, Sergio Escalera, ...


