







## Software

# CAFA-evaluator: a Python tool for benchmarking ontological classification methods

Damiano Piovesan <sup>1,\*</sup>, Davide Zago<sup>2</sup>, Parnal Joshi <sup>2,3</sup>, M. Clara De Paolis Kaluza<sup>4</sup>, Mahta Mehdiabadi<sup>1</sup>, Rashika Ramola<sup>4</sup>, Alexander Miguel Monzon <sup>5</sup>, Walter Reade<sup>6</sup>, Iddo Friedberg <sup>3</sup>, Predrag Radivojac <sup>4</sup>, Silvio C.E. Tosatto <sup>1</sup>

<sup>1</sup>Department of Biomedical Sciences, University of Padova, 35121 Padova, Italy

<sup>2</sup>Program in Bioinformatics and Computational Biology, Iowa State University, Ames, IA 50011, United States

<sup>3</sup>Department of Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, IA 50011, United States

<sup>4</sup>Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115, United States

<sup>5</sup>Department of Information Engineering, University of Padova, 35121 Padova, Italy

<sup>6</sup>Kaggle, San Francisco, CA, United States

\*Corresponding author. Department of Biomedical Sciences, University of Padova, Via Ugo Bassi, 58/B, 35121 Padova, Italy. E-mail: damiano.piovesan@unipd.it

Associate Editor: Cecilia Arighi

## Abstract

We present CAFA-evaluator, a powerful Python program designed to evaluate the performance of prediction methods on targets with hierarchical concept dependencies. It generalizes multi-label evaluation to modern ontologies where the prediction targets are drawn from a directed acyclic graph and achieves high efficiency by leveraging matrix computation and topological sorting. The program requirements include a small number of standard Python libraries, making CAFA-evaluator easy to maintain. The code replicates the Critical Assessment of protein Function Annotation (CAFA) benchmarking, which evaluates predictions of the consistent subgraphs in Gene Ontology. Owing to its reliability and accuracy, the organizers have selected CAFA-evaluator as the official CAFA evaluation software.

**Availability and implementation:** <https://pypi.org/project/cafaeval>

## 1 Introduction

Translating experimental data into biological knowledge remains a slow process despite the rapid accumulation of data in modern biology. Manually curated databases are the primary source of such knowledge due to their thorough standardization of integrated information, often organized into ontological annotations (The Gene Ontology Consortium 2019). The automated prediction of ontological annotations has become widely adopted in knowledge bases. As a result, ensuring a reliable evaluation of the predicted information remains crucial.

The Critical Assessment of protein Function Annotation (CAFA) initiative provides a well-defined framework for managing hierarchical data and independently evaluates Gene Ontology (GO) prediction methods (Radivojac *et al.* 2013, Jiang *et al.* 2016, Zhou *et al.* 2019). Since its first edition, the CAFA experiment has stimulated a number of theoretical studies about GO prediction and its evaluation (Clark and Radivojac 2013, Peng *et al.* 2018).

Despite the significant impact of CAFA, the development of novel function prediction methods suffers from the lack of an easy-to-use tool for internal benchmarking. Existing solutions are problematic due to missing documentation, hampering their maintenance, portability, development, and use by the scientific community. Moreover, these solutions are tailored specifically for GO terms and the CAFA challenge, incorporating numerous hard-coded parameters.

The CAFA-evaluator package addresses these issues by being easy to use and maintain, fully documented, fast, and generic. It can be used with any type of ontology and annotation, and the dataset processing is entirely separated from the evaluation stage. Additionally, the input format is straightforward. The software has been tested against CAFA2 and CAFA3 data, replicating the exact results provided in their corresponding publications (Jiang *et al.* 2016, Zhou *et al.* 2019). CAFA-evaluator has been recently adopted as the official evaluation tool for the CAFA5 challenge hosted on Kaggle.

The CAFA-evaluator software is open source and freely available for download from GitHub and PyPI. The GitHub repository also includes a detailed Wiki section that offers a comprehensive explanation of the algorithm. This Wiki provides valuable insights into the software and offers concrete examples that demonstrate the impact of selecting different parameters during the final evaluation. It serves as a valuable resource for understanding the software and its functionality.

## 2 Implementation

The CAFA-evaluator repository includes a Python library and a user-friendly command-line interface for generating all evaluations and a Python notebook for plotting the results. The evaluation module calculates the *F*-measure, weighted

Received: November 20, 2023; Revised: February 2, 2024; Editorial Decision: March 6, 2024; Accepted: March 12, 2024

© The Author(s) 2024. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<https://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

$F$ -measure, and semantic distance ( $S$ -score), as well as precision–recall and remaining uncertainty–misinformation curves, as described in Jiang *et al.* (2016). The package requires only three standard Python libraries: Numpy, Pandas, and Matplotlib, with the latter being necessary only for generating plots.

## 2.1 Input and calculation

The CAFA-evaluator workflow is shown in Fig. 1. The software requires three inputs: an ontology OBO file, a ground truth file, and the path to the folder containing the prediction file(s). Optionally, it also accepts an information accretion file, which triggers the generation of weighted measures such as weighted precision, recall,  $F$ -measure, and  $S$ -score.

All input files undergo internal parsing, and predictions are filtered to include only those targets present in the ground truth and those terms that are part of the input ontology. When terms are associated with a “namespace,” also called “aspect” or “sub-ontology,” different namespaces are treated as independent ontologies, and both the ground truth and predictions are split accordingly. Namespaces with multiple roots are managed without problems and it is possible to exclude root terms from the evaluation.

The algorithm stores three sparse matrices in memory: the ontology graph as an adjacency matrix, a Boolean  $n \times m$  matrix, where  $n$  is the number of targets and  $m$  is the number of ontology terms, representing the ground truth, and a matrix of the same size (or smaller if some targets are missing) including the prediction scores. Multiple prediction files, each corresponding to a different method, are processed one by one to release the memory associated with the third matrix.

Both the predictions and the ground truth annotations are always propagated up to the ontology root(s). By default, however, prediction scores are propagated without overwriting parents’ scores, as in CAFA. Optionally, the maximum score over all direct children terms can be propagated to their common parent term. The ontology graph is topologically sorted at the parsing time, allowing the propagation to be calculated in linear time, solely depending on the size of the

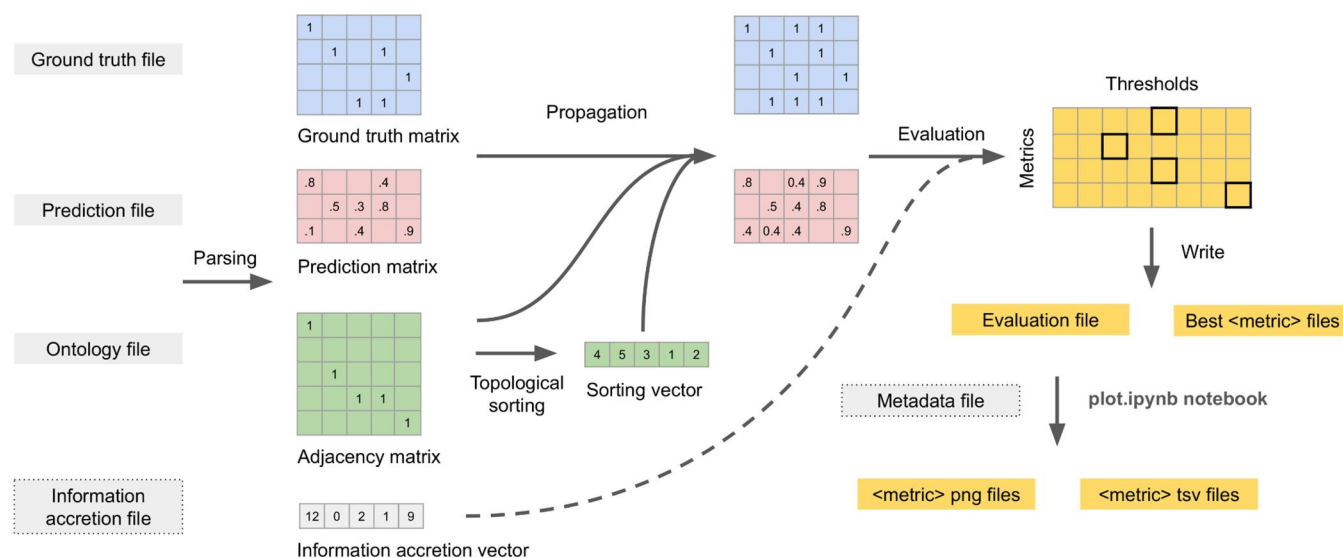
ontology, which is always the same for all prediction files and is loaded in memory at the beginning.

Confusion matrices are calculated per target and per threshold, i.e. separately by considering predicted terms with a score above the threshold. By default, 100 evenly spaced cutoffs in the range [0–1] are considered, but more cutoffs can be set by the user; e.g. to capture all unique score predictions for a method. Calculation time depends on the number of threshold cutoffs. The software is parallelized so that blocks of thresholds can be calculated in different threads.

The tool incorporates both macro- and micro-averaging techniques. The macro-averaging approach follows the traditional CAFA method, where metrics are calculated individually for each target (confusion matrix) and then averaged across all targets. Conversely, the micro-averaging approach involves averaging the confusion matrices over the number of targets before calculating the metrics. These two approaches provide different perspectives on the evaluation process and offer a comprehensive analysis of the software’s performance.

Additionally, the user can decide whether to normalize considering all ground truth targets, i.e. penalizing methods with low coverage, or considering only the predicted targets. By default, the program normalizes the recall by the number of ground truth targets and the precision by the number of predicted targets, as in CAFA.

When the information accretion file is provided, the confusion matrix is calculated after the terms are weighted by their information accretion. This approach avoids returning the simple count as in the confusion matrix when calculating the graph intersection. Other options control the inclusion or exclusion of root (orphan) terms from the evaluation and limit the number of processed terms per protein and namespace. The latter is particularly useful when prediction methods include a large number of predicted terms per target and when the number of targets is large. In any case, the number of considered terms does not affect the computation or memory usage. More information about the impact of the parameters, a detailed workflow of the algorithm along with explanatory



**Figure 1.** CAFA-evaluator workflow. Gray boxes represent input files, while yellow boxes represent output files. Optional input files are outlined with a dashed line. Internal data structures are depicted as matrices and vectors, with example values provided. Arrows indicate logical processes, often corresponding to code functions. At the end of the workflow, image files are generated using a Jupyter Notebook (plot.ipynb), which takes the output of the CAFA-evaluator library as input.

examples are provided in the Wiki of the CAFA-evaluator GitHub repository.

## 2.2 Output

The CAFA-evaluator software generates multiple output objects, including a table with an evaluation row for each method, namespace, and threshold. It also generates an object for  $F$ -measure,  $S$ -score, and weighted  $F$ -measure, reporting the rows with the corresponding best performance. The software also includes a function to store the output into tabular files. Finally, it streams basic execution information, such as timestamps and statistics about the number of processed targets and terms.

The evaluation output table can be used as input for the Python notebook to generate curve plots. The notebook accepts an optional file with the name of the team associated with each prediction file. When this information is provided, only one prediction per team and ontology is selected, as in CAFA. Additionally, prediction files can be associated with a different name, which will be displayed in the plots.

## 3 Summary

The CAFA-evaluator software is an easy-to-use, generic, and well-documented tool designed for benchmarking function prediction methods using any type of ontology and annotation. It requires an ontology OBO file, a ground truth file, a prediction file, and can optionally accept an information accretion file. The software uses internal parsing to filter predictions and generate multiple output files, including a table with an evaluation row for each method, namespace, and threshold, as well as separate files for  $F$ -measure,  $S$ -score, and weighted  $F$ -measure. The software has been tested against CAFA2 and CAFA3 data and has been adopted as the official evaluation tool for the CAFA5 challenge hosted on Kaggle.

## Author contributions

Damiano Piovesan (Conceptualization [lead], Funding acquisition [lead], Methodology [lead], Software [lead], Supervision [lead], Validation [lead], Writing—original draft [lead], Writing—review & editing [lead]), Davide Zago (Investigation [equal], Methodology [equal], Software [equal], Validation [equal]), Parnal Joshi (Methodology [supporting], Software [supporting]), M. Clara De Paolis Kaluza (Investigation [supporting], Validation [supporting]), Mahta Mehdiabadi (Software [supporting], Validation [supporting]), Rashika Ramola (Software [supporting], Validation [supporting]),

Alexander Miguel Monzon (Investigation [supporting], Validation [supporting]), Walter Reade (Investigation [supporting], Validation [equal], Writing—review & editing [supporting]), Iddo Friedberg (Investigation [equal], Methodology [equal], Writing—review & editing [equal]), Predrag Radivojac (Investigation [equal], Methodology [equal], Validation [equal], Writing—original draft [equal], Writing—review & editing [equal]), and Silvio C.E. Tosatto (Methodology [supporting], Validation [supporting])

## Conflict of interest

None declared.

## Funding

This publication was partially based upon work from COST Action ML4NGP (CA21160), supported by COST (European Cooperation in Science and Technology). This work was supported by the European Union through ELIXIR, the research infrastructure for life-science data. NextGenerationEU, PNRR project ELIXIRxNextGenIT (IR0000010) and National Center for Gene Therapy and Drugs based on RNA Technology (CN00000041). Italian Ministry of Education and Research through the NextGenerationEU fund PRIN 2022 project: PLANS (2022W93FTW). Funding for open access charge: University of Padova.

## References

- Clark WT, Radivojac P. Information-theoretic evaluation of predicted ontological annotations. *Bioinformatics* 2013;29:i53–61.
- Jiang Y, Oron TR, Clark WT *et al.* An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biol* 2016;17:184.
- Peng Y, Jiang Y, Radivojac P *et al.* Enumerating consistent sub-graphs of directed acyclic graphs: an insight into biomedical ontologies. *Bioinformatics* 2018;34:i313–22.
- Radivojac P, Clark WT, Oron TR *et al.* A large-scale evaluation of computational protein function prediction. *Nat Methods* 2013; 10:221–7.
- The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acid Res* 2019;47:D330–8.
- Zhou N, Jiang Y, Bergquist TR *et al.* The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol* 2019;20:244.