HFCommunity: A Tool to Analyze the Hugging Face Hub Community

Adem Ait
IN3 – UOC
Barcelona, Spain
aait_mimoune@uoc.edu

Javier Luis Cánovas Izquierdo
IN3 – UOC
Barcelona, Spain
jcanovasi@uoc.edu

Jordi Cabot
IN3 – UOC, ICREA
Barcelona, Spain
jordi.cabot@icrea.cat

***Abstract*—In recent years, empirical studies on software engineering practices have primarily relied on general-purpose social coding platforms such as GITHUB or GITLAB. With the emergence of Machine Learning (ML), platforms specifically designed for hosting and developing ML-based projects have appeared, being HUGGING FACE HUB one of the most popular ones. HUGGING FACE HUB focuses on facilitating the sharing of datasets, pre-trained ML models and applications built with them (*spaces* in HUGGING FACE HUB terminology). Besides, the Hub is adding more and more collaborative features, such as issues and pull requests, to facilitate the building of these artifacts within the platform itself. With over 100K repositories, and growing fast, HUGGING FACE HUB is therefore becoming a promising source of data on all aspects of ML projects and the community interactions around them. As such, we believe it is a promising source for all types of empirical studies aimed at analyzing the collaborative development and evolution of ML artifacts. Nevertheless, apart from the API provided by the platform, there are no easy-to-use solutions to collect and explore the different facets of HUGGING FACE HUB data, including the repositories, discussions and code evolution. To overcome this situation, in this paper we present HFCOMMUNITY, a relational database populated with HUGGING FACE HUB data to facilitate empirical analysis on the growing number of ML-related development projects.**

***Index Terms*—Mining Software Repositories, Data Analysis, Hugging Face**

I. INTRODUCTION

Social coding platforms such as GITHUB, GITLAB, and BITBUCKET have become the *de facto* standard for sharing Open-Source Software (OSS) projects and collaborating on them. These platforms are built on top of Git and rely on the so-called pull-based development model [1], introduced by GITHUB, where developers can create a copy (i.e., fork) of any repository and submit a pull request to the original repository to propose changes. They offer collaboration tools such as issue trackers, discussions and wikis; as well as social features such as the possibility to watch, follow and like other users and projects. Among them, GITHUB has become the largest code hosting site in the world, with more than 80 million users and 200 million repositories.

The emergence of Machine Learning (ML) has triggered the appearance of platforms specifically designed for sharing and developing ML-based projects, being HUGGING FACE HUB one of the most popular ones. HUGGING FACE HUB, initially created as a hosting platform for ML artifacts where people could link related projects and their respective datasets, is moving towards becoming a more collaborative environment with new features (such as discussions support) oriented to increase community interactions. As of November 2022, HUGGING FACE HUB hosts more than 100K repositories, and this number is growing fast.

Its popularity, ML-specific focus and rich metadata make HUGGING FACE HUB a promising source of data for empirical studies on ML development. However, current access to HUGGING FACE HUB repository data is only available programmatically via the official API, which may hamper the adoption and exploration of the different facets of HUGGING FACE HUB data (e.g., the digestion of card data attributes or associations between entities).

In this paper, we present HFCOMMUNITY, a tool that enables researchers to discover HUGGING FACE HUB community insights by collecting and sharing HUGGING FACE HUB data via a relational database. Like existing solutions for general-purpose platforms (e.g., GHTORRENT [2] for GITHUB), HFCOMMUNITY provides domain-specific concepts such as models, datasets, and spaces, thus facilitating its exploration and querying via SQL-like languages.

The rest of the paper is organized as follows. Section II presents the HUGGING FACE domain. Sections III and IV describe the architecture and website of HFCOMMUNITY, respectively. Section V elaborates on the related work. Finally, section VI ends the paper and summarizes the future work.

II. HUGGING FACE HUB

HUGGING FACE HUB is a Git-based social code hosting platform focusing on ML development created by HUGGING FACE, an Artificial Intelligence (AI) company.

Besides providing code-centered features (e.g., commit history via the underlying Git infrastructure), HUGGING FACE HUB promotes collaboration within the repository's community via discussions. Discussions functionality is similar to that of other platforms such as GitHub though with limitations. Under this feature we can find the Q&A oriented threads and the pull request infrastructure. HUGGING FACE HUB repositories can be set to private, ensuring that repository is not visible by other user of the platform, or gated, which require users to agree to share their contact information in order to access.

HUGGING FACE HUB repositories are classified according to three types: (1) models, (2) datasets and (3) spaces. The first two types are related to the development of ML models and datasets, while the third type is related to the development of demo apps for them. The platform offers common features for all types of repositories, such as the possibility of creating, deleting, cloning or uploading files and a natural language description of the artifact (called *card*) but also specific features depending on their type, as we describe in the following.

Model repositories are used to host pre-trained ML models. HUGGING FACE HUB stores information about the dataset/s and library the models rely on, a widget to run inferences for such model, recommended configuration and spaces that use that model for demo applications. Dataset repositories target a variety of datasets useful for training models on tasks such as translation, automatic speech recognition, and image classification. Datasets can be linked to a research article hosted in *Papers with Code*[1] and citation information. For models and datasets, HUGGING FACE HUB also tracks the number of downloads. Finally, space repositories allow developers to easily host ML demo apps on their profile. Spaces aim at showcasing the capabilities of a model and to provide a user-friendly interface to interact with it. Spaces can be statically created via standard Web technologies (i.e., JavaScript and HTML) or rely on SDKs (HUGGING FACE currently supports GRADIO[2] and STREAMLIT[3]).

Another key feature of HUGGING FACE HUB repositories is the repository card commented before, being different for each type of repository. In model repositories, card information is composed by its description, its intended uses and potential limitations, the training parameters and experimental info, which datasets were used to train the model and the evaluation results. In dataset repositories, card information is intended to inform users about how to responsibly use the data and information about any potential biases within the dataset. Furthermore, it is recommended to include dataset metadata, which covers information about a dataset such as its license, language, size and tags, which help users discover a dataset on the Hub. Finally, in space repositories, the card information gives information about the appearance of the repository.

III. ARCHITECTURE

The overall architecture of HFCOMMUNITY is depicted in Figure 1. HFCOMMUNITY collects data from HUGGING FACE HUB and the Git repositories, and stores it in a relational database to facilitate the consumption of project data by data analysts interested in running studies and metrics calculation on the data.

Next we describe in more detail how we designed and populated the database, and how we use it to compute useful metrics.

A. Designing the database

The schema of the relational database is derived from a conceptual schema, expressed as a UML class diagram,

[1]https://paperswithcode.com/
[2]https://gradio.app/
[3]https://streamlit.io/

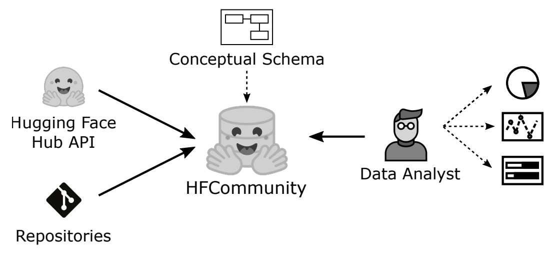


Fig. 1: Overview of HFCOMMUNITY architecture.

modeling the relevant concepts, relationships and properties relevant to the HUGGING FACE HUB and Git domains (see Section II). This conceptual schema is shown in Figure 2.

The schema includes a hierarchy to represent repository types (see `Repository` hierarchy). The common information of any repository (see superclass of the `Repository` hierarchy) is the identifiable data (see `id`, `author`, and `sha`) and its characterization attributes (see `likes`, `private`, `lastModified`, `cardData`, `gated` and `type`). Subclasses of the hierarchy model information specific of each repository type (see `Model`, `Dataset` and `Space` classes). For a model repository we model their type, the widget feature (see `pipeline_tag`), the library used (see `library_name`), and the model configuration information (see `config`). In dataset repositories, there is information about the description of the dataset (see `description`), how to cite the dataset (see `citation`) and the identifier in the PapersWithCode platform (see `paperswithcode_id`). Furthermore, we have the number of downloads of model and dataset repositories (see `downloads`).

Repositories may also include tags and discussion (see `tags` and `discussions` relationships). Discussions have an `id` and include: their title (see `title`), their status (see `status`), when they have been created (see `createdAt`), whether the discussion is a pull request (see `isPullRequest`), and a set of discussion events (see `DiscussionEvent` hierarchy) with additional information for each type of event. These events define the actions in a discussion thread, in particular: the modification of the discussion title (see `DiscussionTitleChange`), which includes the new title (see `new_title`) and the old title (see `old_title`); the modification of the discussion status (see `DiscussionStatusChange`), which includes the new status (see `new_status`); a commit referring the pull request (see `DiscussionCommit`), which includes a summary (see `summary`) and the hash of the commit (see `commit` relationship); and a comment in a discussion (see `DiscussionComment`), which includes the content (see `content`), whether the comment has been edited (see `edited`) and whether the comment has been hidden (see `hidden`).

As HUGGING FACE HUB repositories are based on Git, the schema also models their commit history (see `Commit` class). For a commit, we modeled its creation date and time (see `timestamp`), its author (see `author` relationship), the list

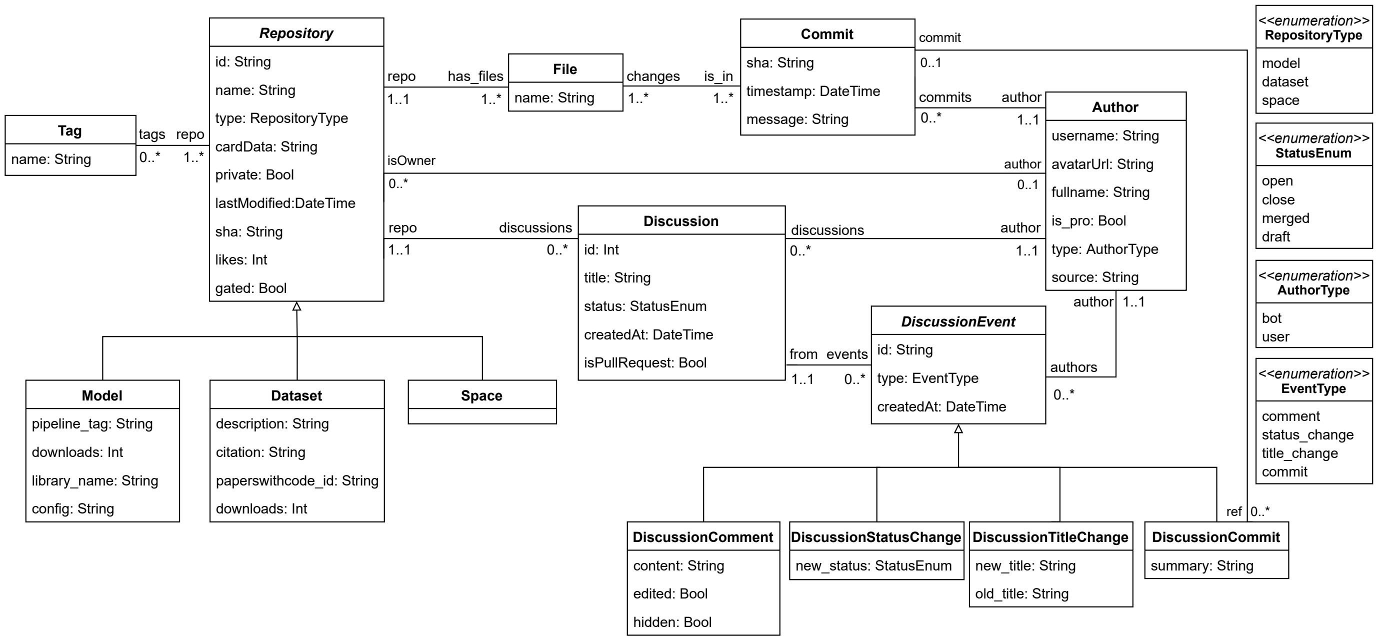


Fig. 2: Conceptual schema of HFCommunity.

of modified files (see `changes` relationship), its message (see `message`), and its hash (see `sha`).

Finally, the schema also includes information about authors of commits, discussions and repository creators (see `Author` class). An author is either a bot or a user (see `type`), and has a name (see `username`), a full name (see `fullname`), the URL of the profile picture in Hugging Face Hub (see `avatarURL`), whether it has a pro account in the Hugging Face Hub (see `is_pro`) and the platform where it belongs, for now being just Git and Hugging Face Hub (see `source`).

Given this UML schema, we generate the structure of tables, columns, keys and foreign keys of our database following the usual mapping rules. In a nutshell, concepts/properties in the conceptual schema are mapped into tables/columns in the database schema, and associations are mapped into foreign keys (e.g., `repo` association between `Repository` and `File`) or new tables (e.g., `changes` association between `Commit` and `File`) depending on the cardinality of the association. To map the `Repository` hierarchy we used the strategy of defining one table with the common attributes of the superclass (`repository`) and one concrete table per each subclass (`model` and `dataset`) with their pertinent attributes plus a foreign key to the table for the superclass. On the other hand, we mapped the whole `DiscussionEvent` hierarchy in a single table (`discussion_event`) having all attributes. The resulting relational schema is available to download [3].

B. Populating the Dataset

To populate the database we first rely on the Hub client library[4]. This is a Python library to facilitate the interaction with the Hugging Face Hub API, such as deleting or cloning a repository, uploading files, and creating and updating a branch. We used this library to collect and fill the database with the Hugging Face Hub information.

Note that there is not always a 1-1 mapping between the API endpoints and our schema, and therefore we had to mix and process the API data to properly populate the tables. In a nutshell, we obtain the list of ML models using the API, and then leverage on other API methods to complement the associations and elements of the schema.

Moreover, we rely on a Git analyzer tool, called PyDriller[5], to recover the data regarding the commit history not available through the Hugging Face Hub API. To optimize the population process, we locally cloned the repositories with the *bare* option, which recovers only the Git administrative metadata and not the actual files. This mechanism skips the downloading of large dataset files, often the case in Hugging Face Hub projects, and allows us to focus on the analysis of the key metadata of the repositories (i.e., commit information).

We run this extraction process on a `MariaDB` database. The populated database is provided as a SQL dump [3]. As of November, 2022, the dataset is formed by 82,357 models, 14,826 datasets and 10,117 spaces.

C. Metric Calculation

From the data available in HFCommunity we can easily query the information and calculate interesting metrics which may help to understand the dynamics of the Hugging Face Hub. As an example, in this section we show a couple of them.

[4]https://github.com/huggingface/huggingface_hub

[5]https://github.com/ishepard/pydriller

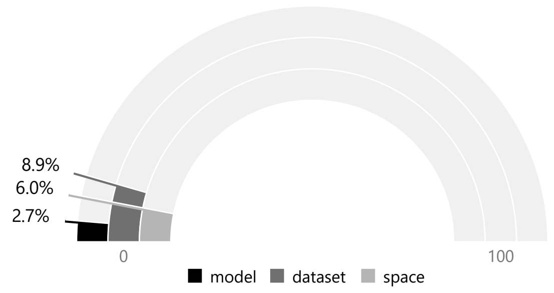


Fig. 3: Example Metric. Percentage of repositories with discussions in HUGGING FACE HUB.

```
SELECT
  COUNT(DISTINCT d.repo_id) AS num_repos, r.type
FROM discussion d
INNER JOIN repository r
ON d.repo_id=r.id
GROUP BY r.type
```

Listing 1: SQL query for metric shown in Figure 3.

For instance, as we already mentioned before, discussions enable the collaboration among all community members, promoting the communication and interaction between contributors. HFCOMMUNITY can be used to measure the usage of discussions according to the repository type. Figure 3 shows the results of this metric and shows that very few repositories leverage this functionality. This metric can be computed thanks to a simple SQL query, shown in Listing 1.

Another example of metric is the number of files in HUGGING FACE HUB repositories. This metric allows us to visualize the typical number of files found in a repository, and also detect empty repositories. Figure 4 shows the results of this metric, and the Listing 2 shows the SQL query. The inner query returns the number of files per each repository, while the outer query return the number of repositories having that exact number of files. As can be seen, more than a half of HUGGING FACE HUB repositories have less than 5 files. Interestingly enough, 14.8% of all repositories in the HUGGING FACE HUB have only 1 file, which may reveals toy or test projects.

IV. TOOL WEBSITE

We have created a website [4] for HFCOMMUNITY as shown in Figure 5. The website is composed of four sections:

1) "Why Hugging Face?", where we give some background about HUGGING FACE;
2) "Information schema", where we describe the conceptual schema;
3) "Download page", where we provide the dump and describe the database design;
4) "Metrics page", where we illustrate the use of HFCOMMUNITY via example metrics as done in Section III-C.

V. RELATED WORK

To collect repository data, there is usually the choice to use an API provided by the hosting platforms itself, such as

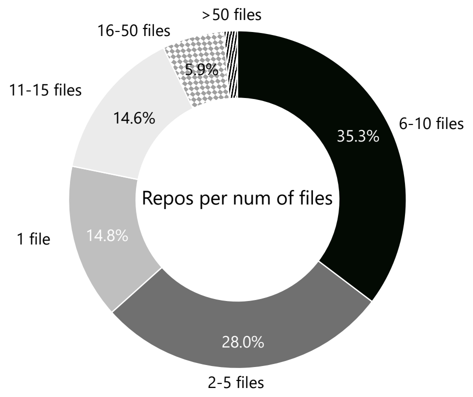


Fig. 4: Example Metric. Number of files in a repository of HUGGING FACE HUB.

```
SELECT files_in_repo AS num_files,
       COUNT(*) AS num_repos
FROM (SELECT repo_id, COUNT(*) AS files_in_repo
      FROM file f
      GROUP BY repo_id) s
GROUP BY files_in_repo
```

Listing 2: SQL query for metric shown in Figure 4.

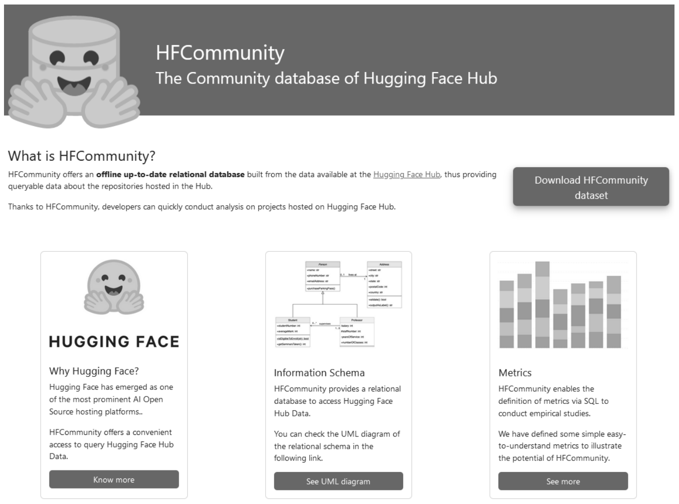
GITHUB which has a REST API[6] and a GraphQL API[7] or HUGGING FACE HUB[8]. There are also tools that are wrappers built on top of such APIs. For instance, GHCRAWLER[9] is a GITHUB API crawler that walks a queue of GITHUB entities transitively retrieving and storing their contents. There are also API crawler tools which target the GITHUB GraphQL API, such as PROMETHEUS by Jobst *et al.* [5].

To facilitate even more the exploitation of the data, several works attempt to collect, digest and publish the data in more accessible formats. Some examples are: GHTORRENT [2], which is a dataset that provides a relational database with the metadata of GITHUB repositories; and GITHUB ARCHIVE[10], which provides access to the metadata of all public events triggered in GITHUB since December, 2011.

While the above stay at the GITHUB level, other works complement them by collecting more fine-grained Git-related activity. PYDRILLER [6] is a Python library that facilitates the extraction of information from Git repositories. GITCOMPARE[11] which extracts some health indicator metrics from a Git repository. GIT2NET [7] is an Open-Source Python package that facilitates the extraction of co-editing networks from Git repositories

Finally, other works take a broader perspective aiming to integrate all types of data sources related to the project. This includes, for instance, discussions, pull requests, conversations from SLACK or DISCORD, emails, etc.). This enables a more

[6]https://docs.github.com/en/rest
[7]https://docs.github.com/en/graphql
[8]https://huggingface.co/docs/huggingface_hub/package_reference/hf_api
[9]https://github.com/Microsoft/ghcrawler
[10]https://www.gharchive.org/
[11]https://gitcompare.com/


HFCommunity

The Community database of Hugging Face Hub

What is HFCommunity?

HFCommunity offers an **offline up-to-date relational database** built from the data available at the Hugging Face Hub, thus providing queryable data about the repositories hosted in the Hub.

Thanks to HFCommunity, developers can quickly conduct analysis on projects hosted on Hugging Face Hub.

HUGGING FACE

Why Hugging Face?

Hugging Face has emerged as one of the most prominent AI Open Source hosting platforms..

HFCommunity offers a convenient access to query Hugging Face Hub Data.

Information Schema

HFCommunity provides a relational database to access Hugging Face Hub Data.

You can check the UML diagram of the relational schema in the following link.

Metrics

HFCommunity enables the definition of metrics via SQL to conduct empirical studies.

We have defined some simple easy-to-understand metrics to illustrate the potential of HFCommunity.

Fig. 5: HFCommunity website.

comprehensive analysis of the development process and community of a project. Some examples are: GrimoireLab [8], SourceCred[12], Augur[13], Kibble[14], or Gitana [9].

Nevertheless, none of the tools above cover Hugging Face Hub. Therefore, to the best of our knowledge, ours is the first tool and dataset aimed at facilitating the analysis of Hugging Face Hub data complementing these previous approaches.

VI. Conclusion

In this paper we have presented HFCommunity, a tool to gather information about the Hugging Face Hub repositories and community discussions. HFCommunity publishes this information as a relational database storing Hugging Face data on repositories, tags, discussions, discussion events, files, commits, authors, etc. The dataset is available for download in the tool website [4].

We believe that the release of HFCommunity opens the door to new empirical studies focused on ML and AI projects, to complement existing literature on the field (e.g., [10]). As any other source of large-scale data, we find crucial to study the perils and promises of mining repositories from Hugging Face Hub, similar to what has been done for other software data sources, see for instance the works by Kalliamvakou et al. [11] and Dabic et al. [12].

The development of HFCommunity is an ongoing work that continuously adapt to the Hugging Face Hub API changes. However, this API still has a number of limitations we had to overcome. First, the API does not provide Git-related data, such as commits, authors, etc, thus limiting the analysis of development practices. We solved this situation by retrieving such information directly from the Git repositories as explained in Section III. However, this introduces a threat to validity, as Git and Hugging Face Hub usernames are not forced to match. Furthermore, the API does not provide information about the creation date of a repository nor you can filter by date when retrieving the projects. At the database level, we take the date of the first commit as creation date but given the growing number of Hugging Face Hub repositories, filtering should be added to the API for scalability reasons. Finally, some valuable repository information is not explicitly available in the API and must be searched within the plain textual description. A clear example is the link to the same repository hosted in GitHub (when exists) which would enable a cross-analysis of the repository data. Another example would be the relationships between spaces, models and datasets that it is also hidden in the card data.

As future work, we plan to perform a NLP-based analysis of the textual repository fields to extract relevant information as the one mentioned above or even more fine-grained annotation data for explainability analysis [13]. Once links between repositories become available (either internal, i.e., among Hugging Face Hub artifacts, or external, i.e., for repos hosted in other platforms) we plan to work on an integration approach that enriches the available project data. Finally, we would like to automate the process to incrementally collect repositories from Hugging Face Hub, thus allowing the dataset to be released more periodically.

[12]https://sourcecred.io/

[13]https://github.com/chaoss/augur

[14]https://kibble.apache.org/

References

[1] G. Gousios, M. Pinzger, and A. van Deursen, "An Exploratory Study of the Pull-based Software Development Model," in *Int. Conf. on Software Engineering*, 2014, pp. 345–355.

[2] G. Gousios, "The Ghtorrent Dataset and Tool Suite," in *Working Conf. on Mining Software Repositories*, 2013, pp. 233–236.

[3] A. Ait, J. Cánovas Izquierdo, and J. Cabot, "HFCommunity Dataset," http://hdl.handle.net/20.500.12004/1/C/SANER/2023/699, 2022.

[4] A. Ait, J. Cánovas Izquierdo, and J. Cabot, "HFCommunity Website," http://hdl.handle.net/20.500.12004/1/C/SANER/2023/218, 2022.

[5] A. Jobst, D. Atzberger, T. Cech, W. Scheibel, M. Trapp, and J. Döllner, "Efficient GitHub Crawling Using the GraphQL API," in *Computational Science and Its Applications*, 2022, pp. 662–677.

[6] D. Spadini, M. Aniche, and A. Bacchelli, "Pydriller: Python Framework for Mining Software Repositories," in *Europ. Software Engineering Conf. and Symp. on the Foundations of Software Engineering*, 2018, pp. 908–911.

[7] C. Gote, I. Scholtes, and F. Schweitzer, "git2net: Mining time-stamped co-editing networks from large git repositories," in *Int. Conf. on Mining Software Repositories*. IEEE Press, 2019, pp. 433–444.

[8] S. Dueñas, V. Cosentino, J. M. González-Barahona, A. del Castillo San Felix, D. Izquierdo-Cortazar, L. Cañas-Díaz, and A. P. García-Plaza, "Grimoirelab: a Toolset for Software Development Analytics," *PeerJ Comput. Sci.*, vol. 7, p. e601, 2021.

[9] V. Cosentino, J. Cánovas Izquierdo, and J. Cabot, "Gitana: a Software Project Inspector," *Sci. Comput. Program.*, vol. 153, pp. 30–33, 2018.

[10] D. Gonzalez, T. Zimmermann, and N. Nagappan, "The State of the Ml-universe: 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub," in *Int. Conf. on Mining Software Repositories*, 2020, pp. 431–442.

[11] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. Germán, and D. E. Damian, "The Promises and Perils of Mining GitHub," in *Working Conf. on Mining Software Repositories*, 2014, pp. 92–101.

[12] O. Dabic, E. Aghajani, and G. Bavota, "Sampling Projects in GitHub for Msr Studies," in *Int. Conf. on Mining Software Repositories*, 2021, pp. 560–564.

[13] J. Giner-Miguelez, A. Gómez, and J. Cabot, "DescribeML: a tool for describing machine learning datasets," in *Int. Conf. on Model Driven Engineering Languages and Systems: Companion Proceedings*, 2022, pp. 22–26.