Supplementary material for: "Renku: a platform for sustainable data science"

Rok Roškar[1], Chandrasekhar Ramakrishnan[1], Michele Volpi[1], Fernando Perez-Cruz[1], Mohammad Alisafaee[2], Philipp Fischer[3], Lilian Gasser[1], Eliza Jean Harris[1], Firat Ozdemir[1], Patrick Paitz[3], Carl Remlinger[2], Luis Salamanca[1], Ralf Grubenmann[1], Tasko Olevski[1], Elisabet Capón García[1], Lorenzo Cavazzi[1], Jakub Chrobasik[2], Andrea Cordoba[1], Alessandro Degano[2], Jimena Dupré[1], Wesley Johnson[1], Eike Kettner[1], Laura Kinkead[1], Seán Murphy[1], Flora Thiebaut[1], Olivier Verscheure[2]

1. Swiss Data Science Center, ETH Zürich, Zürich, Switzerland.
2. Swiss Data Science Center, EPFL, Lausanne, Switzerland.
3. Swiss Federal Institute for Forest, Snow, and Landscape Research, WSL, Birmensdorf, Switzerland

1 Metadata and the Knowledge Graph

1.1 Metadata format

One of the main features of Renku is to record metadata for entities like Projects, Datasets, and Workflows, from the very start of every project, in a machine-readable format. The goal is to capture metadata automatically in a maximally interoperable way so that it can be extended and immediately used for publication, archiving, or by 3rd party systems (such as external repositories or workflow runtimes). Metadata is stored in a hidden directory in each project and users are not expected to manipulate it directly. It can be changed through Renku CLI commands or through the web user interface. If a project is pushed to a Renku-enabled git repository server, the Renku knowledge graph (KG) picks up and indexes the project's metadata in the global KG for that instance.

The primary export format for metadata is RDF (Resource Description Framework)[1], a W3C-recommended model for data exchange on the web. RDF variants, like JSON-LD[2], are used in a variety of applications, including search engine optimization. RDF is defined as a graph of facts expressing relationships between resources by using semantic ontologies to precisely define the meaning of entities and their properties. Different ontologies can be combined in a single RDF graph, meaning that metadata from different domains can be seamlessly joined together.

In accordance with semantic linked data best-practices, we publish the Renku ontology at `https://swissdatasciencecenter.github.io/renku-ontology/`. It is a combination of `schema.org`[3], PROV-O (a W3C ontology for expressing provenance information) and some custom entities and concepts that were lacking in existing ontologies. `Schema.org` provides sufficient expression of concepts for Projects and Datasets, and is largely interoperable with other open publishing metadata standards. This allows us to easily import datasets from external repositories (e.g. *Zenodo*) while ensuring consistency in the metadata. It also allows users to export their datasets at any time, with Renku taking care of most of the metadata exchange with the chosen data repository.

[1]`https://www.w3.org/RDF/`
[2]`https://json-ld.org/`
[3]`https://schema.org`

1.2 The Renku knowledge graph

Metadata is added to Renku projects every time any of the entities in a project are created or updated. If files are added to a Dataset, for example, a new derivative Dataset is created and a pointer is saved to its previous version so that its evolution can easily be tracked. If Renku is used to execute a workflow that tracks inputs and outputs, those are similarly recorded as dependencies and results of an execution plan. The metadata from different entities (e.g. a dataset file can also be an input to a workflow) can easily be combined to form a complete picture of the data provenance.

This integration of metadata about different parts of each project and their relationship to other projects and uses of data, as well as to other entities (e.g. people and organizations) is what forms the Renku KG. An illustration of this structure is shown in Figure 1, which shows an example pipeline from raw data collection to model training. The pipeline spans two different projects (`sentiment-data` and `sentiment-analysis`). In the first project, a workflow first imports data and then transforms it into the `sentiment-data` dataset. This dataset is imported by another project and used to train a model. Each entity has a unique Uniform Resource Identifier (URI) (we do not show the full URI for brevity) and every part of the metadata in the KG has the appropriate semantic type. The KG structure allows us to follow and query such complex scenarios to discover the use of data and the relationships between dataset creators and dataset consumers.

In the current RenkuLab web app implementation, the KG is used for search and for adding additional information to certain entities in the user interface, for example dataset usage across projects. We are currently planning new features to increase the visibility of the KG in future updates.

1.3 Extending the Renku knowledge graph

The Renku metadata in the KG can be extended with custom metadata using plugins for the CLI. Plugins can implement Renku-specific plugin hooks, which will automatically get picked up by the CLI and executed alongside Renku functionality. Besides allowing for customizing and extending Renku behavior for workflow pipelines, datasets and interactive sessions, such as adding new execution backends for workflows, these plugins can also add custom metadata using the `Web Annotation Data Model (OA)`[4]. The OA model allows having arbitrary KG data that points to Renku metadata in the same graph, without having to change the Renku metadata itself. In this way, third parties are free to add domain specific knowledge while keeping the Renku metadata safe and consistent.

2 Compute and Data Access

Using the two different interfaces (Web and CLI), users are free to choose how and where they work with their project. The user should have a uniform experience with regards to compute and data, regardless of their preferred mode of access. Some users prefer using web interfaces, others never leave the terminal; if a platform is to embed itself into the daily workflow, it must strive not to force one mode over others. To make the transition to the cloud even more seamless, users also have the option of taking advantage of remote resources from the comfort of their local IDEs (e.g. *VSCode* or *PyCharm*) by connecting to the session through *SSH*. Local (containerized) or remote sessions can be started from the command line with identical configurations and docker images shared between both.

2.1 Data in Renku

Renku strives to simplify collaborative data access by providing users with consistency regardless of where they work on their project. Data science projects combine code with data to derive their results, and Renku makes choices and provides users options for how to manage these resources. For code, we have chosen git, as the tool for version control. Git is very good at managing text files, but is not always ideal for managing data, which can be large, in a binary format, or both. By default, data is stored using git Large File Storage (git LFS), which enables data to be versioned alongside code. The data itself is not checked-in to the git repository directly; instead, a small text-file pointer is saved and the data is downloaded from and uploaded to another storage location automatically during normal git operations. Git LFS is great for relatively small amounts of data and the Renku

[4]`https://www.w3.org/TR/annotation-model/`

project configuration includes some additional simplifications to make users' experience with git LFS smoother. However, git LFS has some important limitations; once fetched, two copies of the data exist: one in the LFS cache and another in the repository file tree. For large amounts of data, this introduces significant operational storage overhead. Relying on git LFS can be dangerous if large amounts of data are committed directly to the repository accidentally; such mistakes can make repositories unusable and cause much frustration for users.

In cases where larger volumes of data are needed, other solutions such as leveraging cloud storage are required. However, relying on external, decoupled systems introduces barriers for collaborative projects, because it requires managing access in a way that is easily transferable between users. In Renku, Datasets can be configured to use external storage and the details of access (apart from credentials) are stored within the Dataset metadata. Once configured, users do not need to worry about, for example, the data location, its parameters of access or the exact paths. Renku also takes care of either downloading or mounting the data from a remote resource in a way that is backend-agnostic and works transparently regardless of whether the user is using the CLI on a local machine, or running a remote session[5]. There are some trade-offs, though. Git LFS allows data to be versioned and multiple versions to be available. Users of cloud storage need to themselves take care to ensure that they do not overwrite versions of data that they may want to refer to again in the future.

Currently, we support S3 and Azure blob storage, but the system is flexible and can accommodate other storage providers.

3 Deploying Renku

Renku is built as a platform that can be deployed on any standard cloud infrastructure. All that is needed for a standalone deployment is included in the main Renku repository[6] and instructions are provided in our documentation[7]. The basic requirement for the full hosted instance is a vanilla Kubernetes cluster with the ability to provision persistent volumes. Renku can be deployed with its own GitLab instance, or it can be configured to connect to an existing one. It does not require any special privileges in GitLab and works like a standard OAuth application. For authentication, Renku uses *Keycloak* which can also be provided externally.

The public instance at `renkulab.io` is free for all to use with a limited amount of compute resources available. Additional resources can be provisioned on request. Institutions or research groups may prefer an instance of their own, depending on compute, storage, and data security needs. We plan on making it possible to connect external (cloud) resources to projects in the public instance in the future, in line with our efforts to make compute access in general more flexible.

4 Example use-cases

Four of the six projects highlighting the utility of Renku for ML dataset development are already described, together with links, in the main text. Two additional projects are detailed below. All projects are configured to either be used directly from the RenkuLab web interface or, when the resources required are large, they can be run using docker on any other machine. This requires the Renku CLI to be installed[8] and docker to be properly configured. Please refer to the documentation in the individual projects for more specifics.

TimeFRAME: Time-dependent Fractionation And Mixing Evaluation for the interpretation of isotopic datasets Currently available data analysis packages for isotopic timeseries measurements datasets utilise Bayesian hierarchical models to quantify different source contributions in a source mixture, but cannot adequately handle timeseries information, or isotopic fractionation during production and consumption reactions. In the TimeFRAME project, we extended the static FRAME

[5]Flexible data handling in hosted sessions is currently being implemented; feature scheduled for launch in August 2023

[6]`https://github.com/SwissDataScienceCenter/renku`

[7]`https://renku.readthedocs.io/en/stable/how-to-guides/admin/deploying-renku.html`

[8]`https://renku.readthedocs.io/en/stable/how-to-guides/own_machine/cli-installation.html`

approach [3] by using different classes of model to deal with mixing, fractionation and time-dependent data [2].

The Renku platform was used to organise and share data and code in the TimeFRAME project between collaborators during development of the package[1]. Following development, Renku is being used to distribute the model to domain scientists in the isotope measurement community. Renku's user interface allows for clear linkages between the model (R / STAN), the different simulated and real test datasets, and the package documentation (R markdown). TimeFRAME can be accessed by anyone as an R Shiny App on the Renku platform, to allow new users to interactively test the model and understand the different functionalities. For local processing of their own datasets, users can download the Renku repository containing the TimeFRAME R package. The combination of code, data and apps on Renku will enhance the experience of TimeFRAME users and facilitate uptake of the model for isotopic data analysis.

MLED: Machine Learning for Electron Detectors In this project, the researchers particularly valued the Renku user-friendly interface and simple computational resource access, which greatly simplified collaboration with the domain experts. The project aims to predict the exact point of impact of an electron in Hybrid pixel detectors from trajectory images. Hybrid pixel detectors show great potential for use in Electron Microscopy, but poor spatial resolution caused by large pixel size and multiple scattering in the sensor layer inhibiting their application. Different ML algorithms are trained on simulation and measurement data to improve spatial resolution by determining the point where the electron hits the detector and assigning the signal to the correct pixel.

In such an experimental setup, data are regularly updated by e.g. changing the tolerance threshold for noise on the detector images, refining the real electron trajectories, etc. These modifications require the domain and ML experts to iterate rapidly together because the changes have significant effects on the structure of the prediction models and their global objectives that must be understood in the scientific context. The group leverages the easy-to-use interactive compute environments together with data versioning in Renku to enhance their collaboration. The project can be found at [4].

5 Comparison to other systems (details)

Hugging Face The popular ML platform Hugging Face provides tools for collaboration and makes it very easy to share models and datasets. As with Renku, an important feature of the platform is the ability of model authors to link their models to datasets (as long as they are hosted on the same platform). HuggingFace is specifically tied to Python as the programming language and does not offer a way to shed light on the actual process of training the model or building the dataset. Renku lacks the more production- and MLOps-oriented features of HuggingFace.

DVC and DagsHub Data Version Control (DVC) offers a CLI that allows users to manage data in remote storage alongside their code. It is specifically tailored to the ML/data science use case and includes features to simplify the tracking of model training. The DVC ecosystem includes several web applications designed specifically for ML team collaboration. While greatly improving reproducibility of projects, it does not provide tracking of datasets and pipelines across multiple projects and also does not include containerized environments.

DagsHub is a sleek front-end to DVC-enabled data science projects, bundling repository management with experiment tracking and data versioning into one solution. It provides a library to integrate with popular tools like MLFlow. Like DVC itself, DagsHub lacks support for encapsulating the compute environment or linking of datasets or pipelines across projects.

Binder Binder facilitates reproducibility by allowing public git repositories with code in Python, R, and Julia to be run in a containerized environment. Its strength and popularity lie in its simplicity: copy/paste a repository URL and a live compute environment is created for you in a few minutes. For certain situations, this can solve the problem of reproducible environments, but there are limitations (code and data need to be public; GPUs, often needed for ML pipelines, are not available). Renku requires a bit more effort in initial set up, but it then offers greater flexibility in the kinds of computing environments available and supports data reuse and provenance tracking, which are not addressed by Binder.

Google Colab Google Colab provides hosted computing environments supporting zero configuration execution of Jupyter notebooks through their proprietary interface. Collab, like Binder, makes projects more visible and reproducible, neither offers any means of tracking dataset usage within or across projects.

Papers with code The popular *Papers with code* portal provides a link between published papers, datasets, methods, and model performance metrics. Researchers can identify datasets relevant for their work and compare to known performance on specific tasks from the literature. The details of how those datasets were created remain to be specified by the papers' authors and no simple way of verifying that the code runs is provided.

OpenML OpenML builds an ML ecosystem by facilitating the exchange of datasets used for training, and provides tools for ML researchers to define, share, and record various other aspects of the model development lifecycle. For example, users can define *tasks* to be completed on specific datasets and group them in *benchmarks*; details of model training can be encapsulated in *flows* or pipelines, such that others are able to reproduce them; results of training runs can similarly be recorded and shared with the community. In contrast to Renku, OpenML offers features directly tailored to the needs of ML researchers. However, with regards to datasets, they are provided "as-is" (often they seem to be brought into OpenML after having been published elsewhere) and the focus is rather on model training and performance. The benefit of a more generalist framework like Renku is that it allows for an easier cross-over between ML research and domain applications. While OpenML offers an avenue for reproducibility through *flows*, these are fully software-based (not containerized) and focus on the steps in the model training itself, not encapsulating dataset pre-processing and manipulation. Nevertheless, Renku could benefit from some additional views tailored specifically for ML researchers not unlike those found in OpenML, or in the future provide integrations with OpenML.

References

[1] Philipp Fischer. Renku project: N2O Pathway Analysis `https://renkulab.io/projects/fischphi/n2o-pathway-analysis`, 2023.

[2] Philipp Fischer. *Using Bayesian Mixing Models to Unravel Isotopic Data and Quantify N2O Production and Consumption Pathways*. PhD thesis, ETH Zurich, 2023.

[3] Maciej P. Lewicki, Dominika Lewicka-Szczebak, and Grzegorz Skrzypek. FRAME—Monte Carlo model for evaluation of the stable isotope mixing and fractionation. *PLoS ONE*, 17(11 November):4–7, 2022.

[4] Carl Remlinger. Renku project: MLED `https://renkulab.io/projects/carl.remlinger/mled`, 2023.

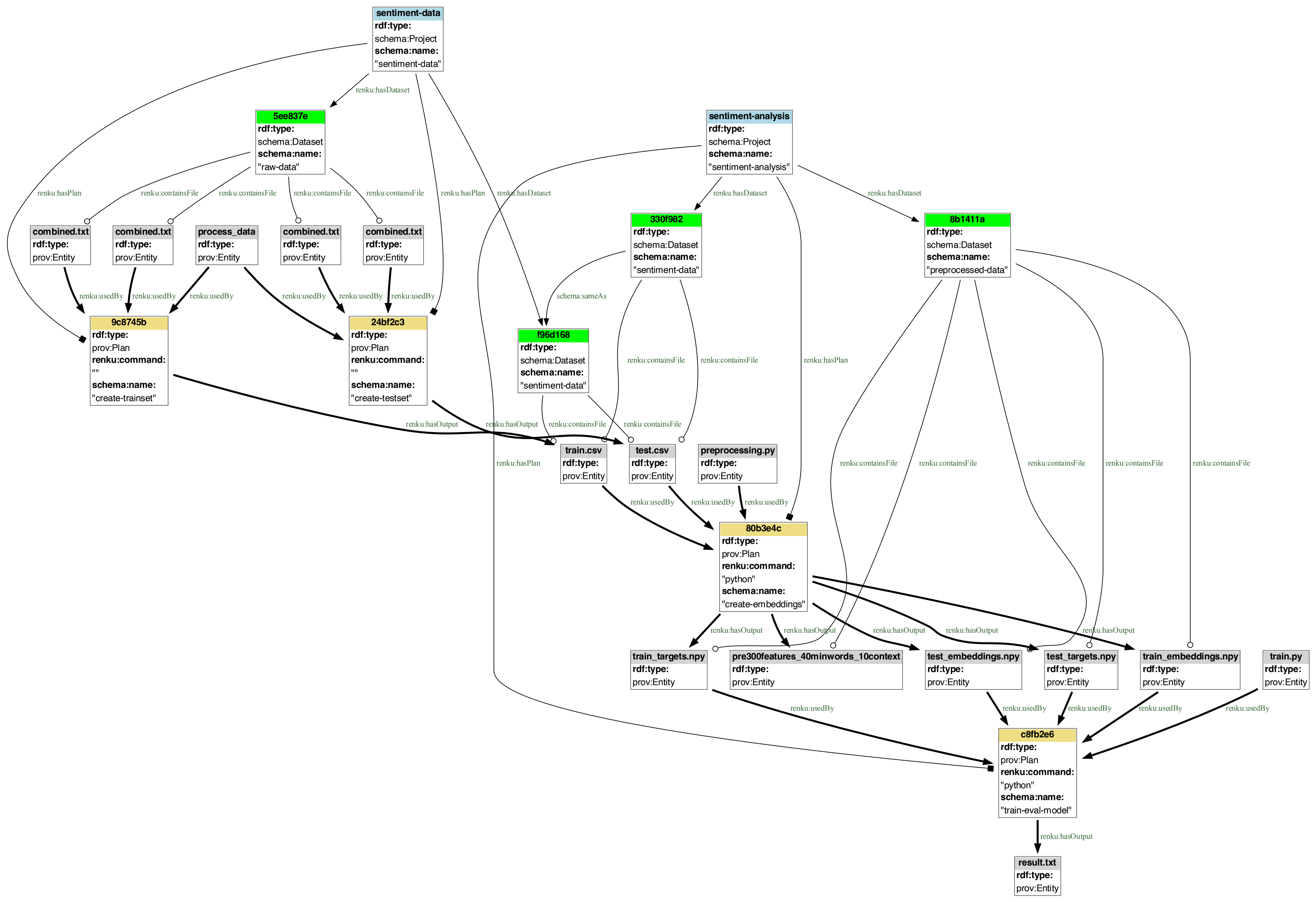

Figure 1: A visualization of the Renku knowledge graph. It contains a full pipeline across two projects. In the first project, a dataset is created and pre-processed. The second project imports the pre-processed dataset and trains a model. The bold arrows indicate the data flow through code executions. Projects are marked in blue, Datasets in green and Workflow steps in yellow. The ontology prefixes are shortened for readability: `schema` for `http://schema.org`, `renku` for `https://swissdatasciencecenter.github.io/renku-ontology/` and `prov` for `https://www.w3.org/TR/prov-o/`.