RESEARCH PAPER

Canonical Workflow for Machine Learning Tasks

Christophe Blanchi[1†], Binyam Gebre[2] & Peter Wittenburg[3]

[1]DONA Foundation, C/O Université de Genève, Rue du Général-Dufour 24, 1204 Genève, Switzerland

[2]bol.com, Papendorpseweg 100, 3528 BJ Utrecht, The Netherlands

[3]FDO Forum, Gemeindweg 55, 47533 Kleve, Germany

Keywords: Workflow; Machine learning; Digital objects; FAIR; Data management

Citation: Blanchi, C., et al.: Canonical workflow for machine learning tasks. Data Intelligence 4(2), 173-185 (2022). doi: 10.1162/dint_a_00124

Received: September 26, 2021; Revised: February 1, 2022; Accepted: February 5, 2022

ABSTRACT

There is a huge gap between (1) the state of workflow technology on the one hand and the practices in the many labs working with data driven methods on the other and (2) the awareness of the FAIR principles and the lack of changes in practices during the last 5 years. The CWFR concept has been defined which is meant to combine these two intentions, increasing the use of workflow technology and improving FAIR compliance. In the study described in this paper we indicate how this could be applied to machine learning which is now used by almost all research disciplines with the well-known effects of a huge lack of repeatability and reproducibility.

Researchers will only change practices if they can work efficiently and are not loaded with additional tasks. A comprehensive CWFR framework would be an umbrella for all steps that need to be carried out to do machine learning on selected data collections and immediately create a comprehensive and FAIR compliant documentation. The researcher is guided by such a framework and information once entered can easily be shared and reused. The many iterations normally required in machine learning can be dealt with efficiently using CWFR methods.

Libraries of components that can be easily orchestrated using FAIR Digital Objects as a common entity to document all actions and to exchange information between steps without the researcher needing to understand anything about PIDs and FDO details is probably the way to increase efficiency in repeating research workflows. As the Galaxy project indicates, the availability of supporting tools will be important to let researchers use these methods. Other as the Galaxy framework suggests, however, it would be necessary to include all steps necessary for doing a machine learning task including those that require human interaction and to document all phases with the help of structured FDOs.

† Corresponding author: Christophe Blanchi (Email: cblanchi@dona.net; ORCID: 0000-0003-2277-5176).

1. INTRODUCTION

A deep analysis of current practices in a variety of data labs [1] showed that in data driven research there are many recurring data processing activities, but that researchers are hardly using workflow technologies. Contributions to the CWFR discussions [2] also indicated that more and more research labs are testing a variety of existing workflow frameworks, such as Galaxy [3] and Jupyter Notebooks [4], for researchers to use. In general, the focus is on isolating the parts of the work that can be executed automatically without human intervention. We call those computational workflows. These unfortunately are only used to describe parts of the activities that researchers are repeatedly performing. Our analysis concluded that there is a lack description of the data sets and the various activities that result in their creation and that there is currently no process place to make them FAIR. In this paper we describe a framework that would address these last two aspects while focusing on the specific example of machine learning. The intention of the framework and its methods are to facilitate the use of workflows by arbitrary researchers and to create FAIR [5] compliant digital entities without adding load to the researcher.

Machine learning (deep learning) is probably one of the most frequently used methods in academia and industry to find hidden patterns in large data sets, combining different modalities and aggregating data from different sources [12]. Its great advantage of not requiring a priori knowledge to be built-in is also its great disadvantage. Large data collections are needed to compute the many free parameters and in supervised training, proper annotations are required. The resulting model then needs to be checked to verify that it will indeed classify the expected patterns correctly and whether, for example, unintended patterns hidden in the data are causing erroneous classifications.

In addition, the usual problems intrinsic to stochastic classification systems have to be dealt with such as, for example, questions as to whether the minimum found during training is close to the absolute minimum, whether there are overspecifications leading to erroneous classifications, whether the coverage of the training set is representative of the set of test cases, etc. In deep learning, where neural networks are covering a multitude of layers and where each layer consists of a multitude of artificial neurons, small variations of initial parameters and the training set will lead to different models each exhibiting slightly different behaviours.

As a result, deep learning involves many training runs each with different initial configurations and training sets until a satisfying result has been achieved. Many iterations are needed, and proper documentation has to be generated to precisely understand what has been tested already and how to interpret the results. In most labs, the current practices do not include any systematic creation and embedding of the needed documentation for each iteration. Documentation, if it is created at all is created in idiosyncratic manners.

Applying canonical workflow methods to such machine learning tasks accompanied with proper documentation makes much sense for all the labs that adopt the data science paradigm. We can distinguish 5 major steps in such work:

(1) Preparation of the concrete machine learning tasks.
(2) Aggregation of suitable collections for training, validation and testing.
(3) Organising and setting up the transformations.
(4) Carrying out the training and validation.
(5) Postprocessing.
(6) Potential re-iterations from earlier steps.

In this paper we will not include the robustness testing phase, since this could be considered as a separate task that could have its own preparation and execution phases and reuse much of the information generated in the previous tasks.

2. SPECIFIC ML ASPECTS

It is important to elaborate on a few aspects specific for machine learning.

2.1 Execution Aspects

At the moment, machine learning jobs are often executed with multilayer networks (beyond 10 layers), and millions of input streams enabling the identification of millions of free parameters that in turn need to be adjusted. Speed is critical during training and most often, states of the model and the process are maintained in ultrafast databases. It is clear to us that creating or updating FAIR Digital Objects (FDO) [6, 7] in this context would interfere with the highly optimised processes of the ML packages and negatively affect performance. As a result, our focus will be on embedding ML packages as actors in the workflow.

State-of-the-art ML Packages are often aligned with solutions such as Hadoop that support efficient parallelisation. By including such packages as actors in a workflow we will not interfere with Hadoop's parallelisation strategies, for example.

2.2 Current Trends in ML

In this paper we will not elaborate on new trends in ML such as (1) "transfer learning" [8] including the challenge of "catastrophic forgetting" [13] and (2) "validation while learning" [14]. "Transfer learning" simply means that one does not start training from scratch if additional data will be included. In such cases the existing model should be taken and extended based on the additional data. One of the challenges is to take measures that the already trained model does not forget all it had learned beforehand when the additional data is added. This can for example be done by restricting the changes of some parameters [13].

Often validation during training is applied to check the behaviour of the network based on a validation data set. Also in this case the suggested CWFR method does not touch details of processing. One would have to add another input FDO—the validation set—and transform all input to the format required by the ML package being used.

2.3 Explainability of the Model

ML has the disadvantage that hidden biases in the training material could let the model learn to react on unintended patterns, i.e., result in a non-intended behaviour. With huge training sets it will be impossible to know exactly all details of the data sets. One way to understand the behaviour of the model is to carry out specific operations after having checked the formal validation based on robustness tests. In this paper we will not include elaborations on this kind of work since it can be seen as a separate phase.

3. WORKFLOW PHASES

3.1 Preparation

The preparation phase of a workflow experiment includes a few steps:

- Specify the typical descriptive metadata: intentions, timing, actors, organisational framework, etc. using standard (DC, CKAN) or domain-wide vocabularies where possible.
- Select a ML Software Package, organising access to compute cycles etc.
- Define typical processing parameters for proper documentation such as parameters for the ML program (model (layers, terms, connectivity), error, step size, max. iterations, etc.).

To carry out this work typically editors are being launched that support the requested document format, controlled vocabularies and constrained input fields resulting in FAIR compliant descriptions based on widely agreed standards. We see two FDOs as the result of this preparation step: (1) An Exp_Meta_FDO (Figure 1) is being created that has as bit sequence a complete description of all information summarised in the first two bullet points. (2) An Exp_ML_FDO that includes all parameters to control the execution of the ML training task given a specific ML package. To maintain relevant relations Exp_Meta_FDO includes a PID referring to Exp_ML_FDO. Obviously Exp_ML_FDO can be reused in multiple experiments and thus needs to stand alone. The meta data of both FDOs includes typical information about findability and can partly be created automatically from the included bit-sequences[①].

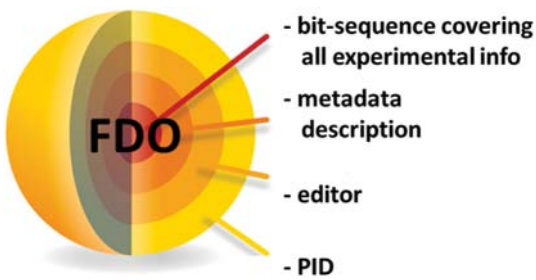

Figure 1. The graphic representation of an FDO as used within the FDO community with a bit-sequence (here capturing all experimental information), associated metadata, a reference to an editor operating on the metadata and a PID referencing to the FDO.

[①] In this paper we will not create a separate FDO for the metadata, but expect it be stored in some database.

Creating FDOs makes both entities FAIR compliant and 1st class citizens in the Internet, i.e., they can be stored, exchanged and reused in different circumstances.

3.2 Collection Aggregation

In this phase the collection needs to be created that is used during training. To carry out this step a collection builder would be selected that creates an FDO as result that encodes the collection in a standard manner as for example suggested by the RDA Collection Group.

- A suitable collection builder is started.
- Within the collection builder a search is being started using some metadata portals.
- The search results are being scanned and suitable files are being selected and added to the collection, their accessibility needs to be checked (correct reference, license, etc.).
- The last two steps can be repeated of course to add further files.
- The collection builder creates a FAIR Digital Object (FDO) which includes
  - some descriptive metadata incl. a Handle
  - content that contains all references which can be URLs to files (all in a distributed scenario).
- The collection builder can be asked to create a "local collection" which means to download all files, put them on a local store for efficiency reasons and add machine actionable references to the collection content.
- Finally, the collection builder creates the FDO which has some metadata, assigns a PID and has as body the references to all included data files.

The interface to other steps is the FDO which we call Exp_Col_FDO (Fig. 2). It includes the collection in a standard format and associated metadata which eventually contains a reference to a local store. The collection can include references of files that are stored locally or in remote repositories. Exp_Col_FDO can be opened again at any time with a suitable collection builder to modify its content, to add new files by starting a new search, etc., i.e. the researcher can quickly change the training set which is crucial for the results to be achieved. In each case a new Exp_Col_FDO assigned with a new PID is being created. Also Exp_Col_FDO is FAIR compliant and as another 1st class citizens in the Internet can be exchanged and reused by different groups with similar purposes. In addition, it is clearly documented by this FDO which data is being used in which order.

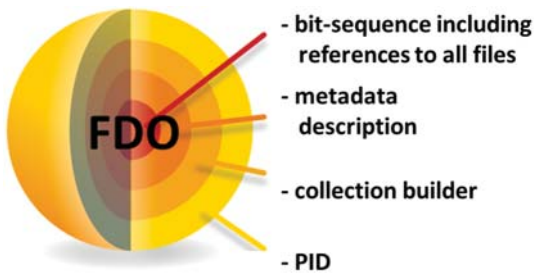

Figure 2. The process how the collection FDO is being configured. It's bit-sequence consists of all the references to the parts of the collection in some structured way. It has a metadata description. One of its operators is the collection builder and it is represented by a PID.

3.3 Organising Transformations

In this phase we need to integrate the digital objects we already have and carry out some typical orchestration work, i.e., transform all information in a way that is required by the ML software package so that the execution of training can be done efficiently.

The canonical workflow requires to select one of the Exp_Meta_FDO that have been created and one of the collections that have been built virtually yielding an Exp_Col_FDO. The Exp_Meta_FDO description needs to be extended to include a reference (PID) to the Exp_Col_FDO and eventually parameters in the experiment description such as number of files included need to be updated.

Since the ML package will have specific requirements about how parameters and data need to be presented some transformations may need to be carried out. Given the large number of different packages it is not yet clear in how far this step can be simplified for the end users. Ideally a set of converter routines would be available that transform the canonical and standardised formats embedded in FDOs into the formats required by the ML package[2]. In some cases ML packages expect to get a path to a directory structure to extract the files, in other cases simple lists are required. Whatever the required formats are, if new data is added or if additional references are required, these should be added into Exp_Meta_FDO to have a complete description.

This step contains what often is called orchestration, i.e., identifying the actors involved in a workflow and creating the output-input binding including conversions etc. In many workflow packages specific graphic tools are supporting orchestration. In canonical workflows this is just a fragment of the whole workflow process. At the end a WF-FDO is being created that contains the sequence of steps and the actors involved. Exp_Meta_FDO is extended to include the link.

3.4 Training

In this phase all relevant information extracted from the FDOs (experiment, collection, machine learning) as required by the ML package are being made available to a wrapper that embeds the ML package. All information needed including references is in the meta FDO which serves as a kind of anchor.

After having carried out all adaptations the ML training run will be started by a call which will include all needed information as expected by the software[3]. On convergence the ML package will terminate and return control to the wrapper. The wrapper receives the output information from the ML package which especially consists of the trained model and a few parameters such as number of iterations, remaining error, validation error, etc. To cover these data an additional FDO is being created which we call Exp_Mod_FDO

[2] We ignore here the aspect that in some cases the format of the data files need to be adapted. This would require other converters and provenance entries in Exp_ML_FDO.

[3] In this paper we will not discuss practices where during training already tests will be carried out to save CPU time when no satisfying convergence can be achieved. This would require additional structures to be offered.

(Figure 3). It contains the complete model in a standard format as its bit-sequence, some metadata (link to the Exp_Meta_FDO, creation time, machine used, ML package used, etc.) and is assigned a PID. In addition, the Exp_Meta_FDO is extended by a link to this new FDO. In some cases the resulting Exp_Mod_FDO might also contain information about the system resources being used which is important for accounting purposes. If this is included in a separate file then another FDO should be created and Exp_Meta_FDO needs to be extended by another attribute. In this paper we will ignore such details.

The wrapper finishes the training phase by returning control to the canonical workflow process.

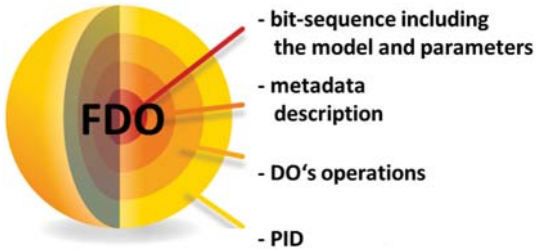

Figure 3. The FDO containing the MLmodel is being configured. It is bit sequence includes all model parameters.

3.5 Postprocessing

In general, some postprocessing is being done at the end of this experiment to document what has been done. It should be noted that Exp_Meta_FDO includes a full description of the whole experiment and includes references to all relevant FDOs such as Exp_Mod_FDO and Exp_ML_FDO. In some labs it might be requested to create semi-formal PDFs or a lab-notebook entry to document what has been done and some findings. Some software which we simply call postprocessor needs to be developed and integrated that turns FDO-based content into the required output formats.

4. FINAL CANONICAL WORKFLOW AND VRE

Here we want to sketch the final canonical workflow that would be implemented making use of canonical components and using FDOs as the combining glue (Figure 4). We ignore side aspects such as starting a timer when starting the training which allows the user to set a maximal time and then to stop the training process. We assume anyhow that the workflow framework (language and execution) will enable asynchronicity which also includes steps where user interactions will lead to waiting times etc.

For all steps we assume the availability of appropriate components (tools) that can be integrated into the workflow and that support the FDO types that are being created and the structure of which needs to be highly described by widely accepted standards to enable sharing and reuse. We also assume that the components are registered in an open software library from which instances can be included in workflows.

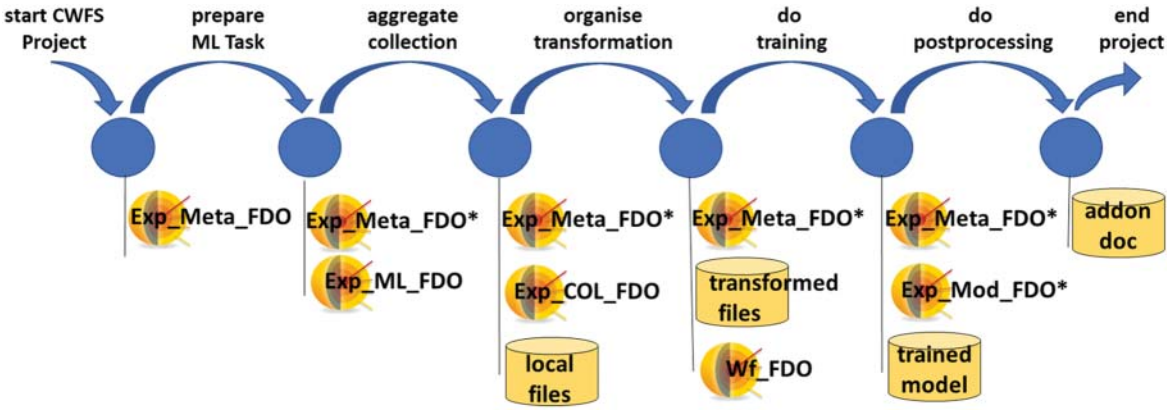

Figure 4. The resulting canonical workflow for a typical ML task including the important FDOs and files that are being created and used. We are using here the graphical notation as introduced by the CWFR initiative.

We assume that the user has a VRE that allows him/her the following actions (Figure 5):

- To create a workflow skeleton (the sequence of actions) by selecting actors per step from a selected library with FDO compliant components;
- To initiate a new experimental project by creating a new Exp_Meta_FDO or (2) start from an existing one by selecting an existing Exp_Meta_FDO; and
- To continue stepwise the sequence of operations by creating the described FDOs and where applicable other entities.

It should be noted that this canonical workflow guides users through the whole process and documents all steps in a comprehensive and FAIR-compliant way applying open standards where possible. At any operation the user can stop and later return to any step to return the process. This canonical workflow is different to the classical workflows which are not meant for human intervention. However, users who would like to change ML parameters or change the collection of files to be used, could simply change appropriate fields and restart the workflow now without any human intervention since all steps including the transformations have been specified.

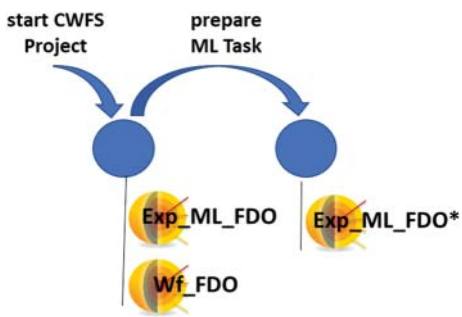

Figure 5. The workflow graph during the orchestration work.

It should also be noted that this canonical workflow does not interfere with the ML software packages which are in general highly specialised tools tuned for processing on special parallel computer architectures. Such a tool is simply embedded with the help of a wrapper.

5. USE OF DOIP

Although the Digital Object Interface Protocol (DOIP) [9] is an efficient protocol to interact with FDOs relevant for all described phases, we will restrict ourselves to describe DOIP's usage in the preparation phase.

This first step after the initialisation of the project is a comparably simple case. We assume that

- at the start a first metadata object has been created that contains the obvious choices such as project name, start date, name of the project owner etc.;
- widely accepted vocabularies such as from DCAT are being used where possible and that new categories are being registered to create a machine actionable FDO;
- a repository can act as a DOIP server, i.e., it has at least a software component that translates DOIP calls into internally used structures; and
- the VRE client has the rights to operate on these FDOs.

The left diagram in Figure 6 indicates the actions using a physical layout which could be misinterpreted which why we added the right diagram that takes an object view. A Digital Object is referenced by a PID which has a record associated with useful state information (green colour), it has a bit-sequence encoding its content (grey) and it has a metadata description (yellow). This diagram gives the impression (for drawing simplicity) that the PID record is residing in the repository which is not correct, since it is residing in the PID resolution system. But for our elaborations this can be ignored. For our further elaborations we will switch to a more abstract illustration which is shown in the right diagram.

The interaction/processing steps look as follows:

(1) The client has the PID1 of the metadata object version 1 and requests the bit-sequence of that FDO which contains the metadata specifications.

Metadata1 <= DOIP (PID1, GetMetadataDE(JSON))

(2) The client receives the bit-sequence in JSON format. It opens a useful editor to allow editing the metadata descriptions and to add for example a prose description about the intentions etc. We have a new metadata description.

(3) The client sends the new metadata description to the repository to create a new FDO with metadata version 2 and a new PID2. It must be an operation that automatically creates a reference to the version X description.

PID2 <= DOIP (PID1, CreateDerivedObject(Metadata2))

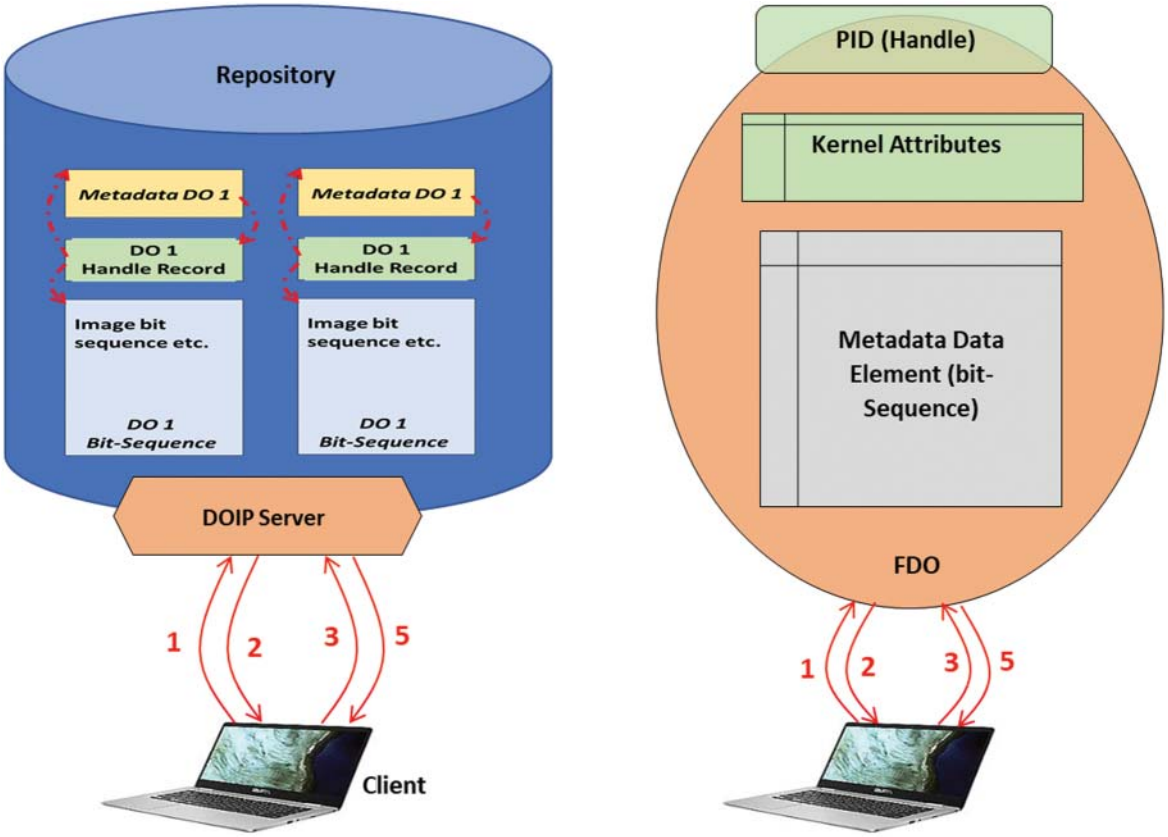

Figure 6. Two complementary parts. The left part indicates the process steps with a traditional repository view in mind. The right part takes a pure object related view. Both together may explain the functioning of the DOIP protocol.

(4) The CreateDerive operation will request a new PID, build the associated PID record, store the new metadata and add a reference to metadata1 using PID1.
(5) The repository returns PID2 of metadata version 2.
(6) The client maintains a history of the process allowing the user to go back to every step and may store it as a separate project FDO.

From this example it is obvious that we need a few primitive operations to deal with a metadata object, some of which are common to all kinds of FDOs:

GetMetadataDE(JSON) get the bit-sequence of a the metadata object, return it in some syntax
CreateMetadataDerive(PID, metadata) create a new metadata object returning a new PID, using metadata as input
GetKernelAttributes(PID) get the kernel attributes of a given PID

GetTypeAttribute(PID) get the type attribute of a given PID
GetKernelProfile(PID) get the used kernel profile (the list of supported attributes)

More primitive operations might be needed, but a balance is needed to restrict the number of operations.

The example above shows one possible implementation. Other implementations can be thought of where for example all intelligence is put into the operations using FDO information and creating new FDOs. In such a scenario the client would be widely simplified, however, for different situations specialistic operations would have to be developed. Therefore, a balance must be found between putting special intelligence into the client and thus reducing the number of operations to be called by DOIP on the one hand and removing intelligence from the client and creating a larger number of operations dealing with all requirements on the other hand. Yet it is too early to make statements.

6. CONCLUSIONS

Workflows and workflow technology are not new and many aspects have been addressed such as workflow languages from BPEL [10] to CWL [11], or frameworks such as Galaxy or Jupyter Notebook as recent developments. Many more can be mentioned as have been mentioned in CWFR meetings for example. Despite these technologies and some exceptions as in bioinformatics for example where standard workflows based on Galaxy are now used by many researchers, we recognised a huge gap between (1) the state of workflow technology and the practices in the many labs working with data driven methods and (2) the awareness of the FAIR principles and the lack of changes in practices during the last 5 years. This gap has been shown by Jeffery et al. [1] when they deeply analysed about 75 research infrastructure reports and proposals.

We need to bring technology even closer to the researchers and reduce the extra effort that is needed for them to apply such advanced technologies. Processes in data-driven science are highly repetitive, often many iterations need to be done to come to satisfying results which is especially true for applying deep learning using large data collections.

In this paper we applied the principles of canonical workflows with FAIR compliant components creating FDOs to indicate a way out of the paradox situation. We assume the existence of libraries of components a researcher can chose and bring together in a sequence of steps. We also assume that all these components create and use FDOs with clearly specified structure and defined semantics. In such cases the user does not have to understand anything about PIDs and FDO details, since tools would become available that are implementing these standards and present details with easy-to-use frontends.

The Galaxy framework is an excellent step towards helping researchers in using automatic frameworks. Yet it is only focusing on computational workflows, i.e., it does not include steps with human interaction, and its comprehensive documentation is captured in a complex database. Including human interactions and creating FDOs as standardised information sources about all relevant aspects would come close to what CWFR intends.

The price the users would have to pay is that they would create many FDOs requiring an intelligent VRE frontend to support easy navigation.

AUTHOR CONTRIBUTIONS

C. Blanchi (cblanchi@dona.net) is core developer of components of the Digital Object Architecture and therefore a major contributor to the paper. B. Gebre (bgebre@bol.com) is specialist for advanced Machine Learning applications in industry and major contributor to those aspects. P. Wittenburg (peter.wittenburg@mpcdf.mpg.de) is co-editor of the FAIR principles paper and member of the FDO Forum and major contributor to the FAIR and FDO aspects.

REFERENCES

[1] Jeffery, K.G., et al.: Not ready for convergence in data infrastructures. Data Intelligence 3(1), 116–135 (2021)
[2] Hardisty, A., Wittenburg, P. (eds.): Canonical Workflow Framework for Research (CWFR)—position paper—version 2, December 2020. Working paper. Available at: https://osf.io/9e3vc/. Accessed 9 December 2021
[3] Welcome to the Galaxy community hub. Available at: https://galaxyproject.org/. Accessed 25 September 2021
[4] Jupyter Notebook. Available at: https://jupyter.org/. Accessed 25 September 2021
[5] Wilkinson, M., et al.: The FAIR guiding principles for scientific data management and stewardship. Scientific Data 3, 160018 (2016)
[6] FDO Forum. Available at: https://fairdo.org/. Accessed 25 September 2021
[7] de Smedt, K., Koureas, D., Wittenburg, P.: Analysis of scientific practice towards FAIR digital objects. Available at: http://doi.org/10.23728/b2share.e14269d07ce84027a7f79ee06b994ef9. Accessed 25 September 2021
[8] Weiss, K., Khoshgoftaar, T.M., Wang, D.: A survey of transfer learning. Journal of Big Data 3, Article No. 9 (2016)
[9] DONA Foundation. Digital object interface protocol specificiation—version 2.0, November 12, 2018. Available at: https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf. Accessed 25 September 2021
[10] Business process execution langauge. Available at: https://en.wikipedia.org/wiki/Business_Process_Execution_Language. Accessed 25 September 2021
[11] Common workflow lanague. Available at: https://en.wikipedia.org/wiki/Common_Workflow_Language. Accessed 25 September 2021
[12] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521, 436–444 (2015)
[13] Kirkpatrick, J., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences 114(13), 3521–3526 (2017)
[14] Sculley, D., et al.: Hidden technical debt in machine learning systems. In: Advances in Neural Information Processing Systems (NIPS 2015), pp. 2503–2511 (2015)

AUTHOR BIOGRAPHY

Christophe Blanchi has background in mechanical engineering and years of experience in computer science, system architecture, distributed systems, computer security, and information management. As a computer scientist at CNRI he focused on researching and developing the Digital Object Architecture and its various systems and components across a wide range of projects within public and private organizations. Christophe Blanchi is currently Executive Director at the DONA Foundation in Geneva where he is responsible for its day-to-day operations, evolving the Digital Object Architecture related standards, ensuring the proper operations of the Global Handle Registry in collaboration with Multi-Primary Administrators across the world, and promoting the adoption of the Digital Object Architecture. It is in this role that Christophe Blanchi is active in the FAIR Digital Object Forum community.
ORCID: 0000-0003-2277-5176

Binyam Gebre has background in computer science and artificial intelligence. He has been working with diverse industry applications involving machine learning, natural language processing and computer vision systems. He has worked as a deep learning scientist for Philips Research in the areas of medical image understanding and personal health. Currently, Binyam is working as a senior data scientist at bol.com, the largest e-commerce company in the Netherlands. He is responsible for building large-scale recommender systems using deep learning.
ORCID: 0000-0003-2656-2055

Peter Wittenburg has a background in electrical engineering, has been working as Technical Director at the Max Planck Institute for Psycholinguistics for many years and acted as member of the IT Advisory board of the president of the MPS. The Max Planck Institute was from the beginning focusing on digital technologies to understand the functioning of the brain with respect to language processing. The institutes need for getting access to data from other institutes to feed the stochastic engines they applied rather early led him to become an expert in building data/research infrastructures. He was responsible for the technological aspects of three large international and European research infrastructures: DOBES, CLARIN and EUDAT. In this function he understood that data work across silos is highly inefficient and that harmonisation and standardisation is required to improve the situation. This was the reason that he co-founded the Research Data Alliance in 2013 and the FAIR Digital Object Forum in 2019.
ORCID: 0000-0003-3538-0106