ML-Schema: An interchangeable format for description of machine learning experiments

Gustavo Correa Publio [a,*], Agnieszka Ławrynowicz [b], Larisa Soldatova [c], Panče Panov [d], Diego Esteves [e,h], Joaquin Vanschoren [f], Tommaso Soru [g]

[a] *Institut für Informatik, AKSW Group, Universität Leipzig, Germany*
E-mail: gustavo.publio@informatik.uni-leipzig.de
[b] *Faculty of Computing, Poznan University of Technology, Poland*
E-mail: agnieszka.lawrynowicz@cs.put.poznan.pl
[c] *Department of Computing, Goldsmiths, University of London,United Kingdom*
E-mail: l.soldatova@gold.ac.uk
[d] *Department of Knowledge Technologies, Jožef Stefan Institute,Slovenia*
E-mail: pance.panov@ijs.si
[e] *SDA Research, University of Bonn,Germany*
E-mail: esteves@cs.uni-bonn.de
[f] *Mathematics and Computer Science, Eindhoven University of Technology,Netherlands*
E-mail: j.vanschoren@tue.nl
[g] *Research Group, Semantic Integration Ltd.,United Kingdom*
E-mail: tom@tommaso-soru.it
[h] *Machine Learning Department, Fraunhofer,Portugal*
E-mail: esteves@cs.uni-bonn.de

Abstract. In this paper, we present the ML-Schema, proposed by the W3C Machine Learning Schema Community Group. ML-Schema is a top-level ontology that provides a set of classes, properties, and restrictions for representing and interchanging information on machine learning algorithms, datasets, and experiments. ML-Schema, a canonical format, resulted of more than seven years of experience of different research institutions. We discuss the main challenge in the development of ML-Schema, which have been to align existing machine learning ontologies and other relevant representations designed for a range of particular purposes following sometimes incompatible design principles, resulting in different not easily interoperable structures. The resulting ML-Schema can now be easily extended and specialized allowing to map other more domain-specific ontologies developed in the area of machine learning and data mining.

Keywords: ontology, data interchange standard, machine learning

1. Introduction

Machine learning (ML) experiments are complex studies involving many steps and iterations requiring expert knowledge. Ensuring that ML research outcomes are properly comparable, understandable, interpretable, reusable and reproducible is a challenge that many proposals, such as *Wings* [1], *OpenTox* [2] and *MyExperiment* [3], have tried to address. Nevertheless, each of them deals with a set of specific scenarios, and fails to address a broader, generic approach in the context of reproducible and reusable science.

Ontologies, as formal machine readable knowledge representations, have the potential to help achieve this goal. An ontology formally defines essential concepts, their properties, and relevant axioms pertinent to a particular area of interest [4].

*Corresponding author. E-mail: gustavo.publio@informatik.uni-leipzig.de.

Recently, several ontologies have been proposed to model the area of machine learning. *Onto-DM* (an Ontology of Data Mining) provides generic representations of principle entities in the area of data mining [5]. *DMOP* (Data Mining OPtimization ontology) has been developed to support meta-mining, i.e. meta-learning from complete ML processes [6]. *Exposé* has been designed to describe and reason about ML experiments [7]. It underpins *OpenML* [8], a collaborative meta-learning platform for machine learning [8]. Finally, the *MEX Vocabulary* (composed of mex-core, mex-algo and mex-perf) aims to tackle the problem of managing ML outcomes and sharing provenance information, particularly on the basic ML iterations, in a lightweight format [9].

The development of these ML ontologies is a significant step towards ensuring unambiguous interpretability and reproducibility of ML experiments. However, none of the existing ontologies fully covers the area of machine learning and supports all the needs for the representation and encoding ML experiments. Instead of the development of a comprehensive general purpose ML ontology, here we propose a more practical and flexible approach that involves the development of ML-Schema – Machine Learning Schema (MLS) – for mapping of the existing ML ontologies and to support a variety of useful extensions. To achieve this ambitious goal, in September 2015 developers of several ML ontologies formed a W3C Community Group[1]. The development of MLS has been initiated as an attempt to prevent a proliferation of incompatible ML ontologies and to increase interoperability among existing ones. The *MLS Community Group* (MLS-CG) is an open-source community comprehending over 50 international researchers.

The main challenge in the development of MLS is to align existing ML ontologies and other relevant representations designed for a range of particular purposes following sometimes incompatible design principles, resulting in different not easily interoperable structures. Moreover, ML experiments are run on different ML platforms; each of those having specific conceptualization or schema for representing data and metadata.

To address the challenge, the members of the MLS-CG identified and aligned a set of principle ML entities – a core ML vocabulary. The core vocabulary of MLS deals with ML algorithms. The schema is focusing on the representation of the algorithms, the machine learning tasks they address, their implementations and executions, as well as inputs (e.g., data), outputs (e.g., models), and performances. The schema also defines a relationship between machine learning algorithms and their single executions (runs), experiments and studies encompassing them.

The terms in the core vocabulary were defined and manually mapped to the ML ontologies participating in this endeavor through several rounds of consultations. In 2016, the MLS-CG published an online proposal for MLS, and welcomed comments and suggestions from the research community and wider [10].

In this paper, we present the results of three years of MLS-CG efforts in standardization of the encoding of ML experiments. MLS aims to support a high level of interoperability among scientific experiments concerning machine learning to foster reproducible research. MLS enables recording of machine learning studies and results as linked open data. MLS is beneficial to ML experiments Ecosystems (e.g., OpenML and *Research Objects* [11]) and ML Metadata Repositories (e.g., *WASOTA* [12]) by providing a more representative standard for their architectures. In OpenML, MLS is used to export all machine learning datasets, tasks, workflows, and runs as linked open data. This allows scientists to connect the results of their machine learning experiments to other knowledge sources, or to build novel knowledge bases for machine learning research.

2. The ML-Schema

In this section, we introduce the MLS ontology w.r.t. its aims and design principles. We also describe the properties defined within the MLS ontology namespace.

2.1. The MLS ontology

The main aim of MLS (or the MLS ontology) is to provide a high-level standard to represent ML experiments in a concise, unambiguous and computationally processable manner. In particular, it aims to align existing ML ontologies and to support development of more specific ontologies for particular purposes and applications.

To serve its purposes, MLS ontology has to be compact but sufficiently comprehensive and easily extendable. To achieve such an aim, we chose to design MLS

[1] See www.w3.org/community/ml-schema/

ontology as a light-weight ontology that can be used as a basis for ontology development projects, markup languages and data exchange standards. We then show how the MLS ontology is open for further extensions and mappings to other resources.

For example, MLS ontology can support vertical and horizontal interoperability across various ML environments. Different ML platforms have different underlying schemes for representing data and metadata (see Figure 1: items 3 and 4: vertical interoperability). This may lead to an extra coding-effort (see Figure 1: item 2) if to achieve both the desired interoperability and a better provenance level as well as a more automatized environment for obtaining the generated results. To reduce the gap, ML vocabularies and ontologies have been proposed (see Figure 1: item 5).

The gap can be further significantly reduced by achieving interoperability among state-of-the-art (SOTA) schemata of those resources (see Figure 1: item 5) i.e. achieving the horizontal interoperability (Figure 1: item 6). Therefore, different groups of researchers could exchange SOTA metadata files in a transparent manner, e.g.: from OntoDM and MEX (`MLS.Schemadata=MLS.convert('myfile.ttl',MLS.Ontology.OntoDM,MLS.Ontology.MEX)`).

2.2. MLS Ontology properties

In the following we list and briefly describe the properties modelled in MLS:

- ***achieves***: A relation between a run and a task, where the run achieves specifications formulated by the task.
- ***definedOn***: A relation between a task and either the data or an evaluation specification pertinent to this task.
- ***defines***: The inverse relation of ***definedOn***
- ***executes***: A relation between a run and an implemantation that is being executed during the run.
- ***hasHyperParameters***: A relation between an implementation of a machine learning algorithm and its hyperparameter..
- ***hasInput***: A relation between a run and data that is taken as input to the run.
- ***hasOutput***: A relation between a run and either a model or model evaluation that is produced on it's output.
- ***hasPart***: A relation which represents a part-whole relationship holding between an entity and its part.
- ***hasQuality***: A relation between entities and their various characteristics.
- ***implements***: A relation between an information entity and a specification that it conforms to.
- ***realizes***: A relation between a run and an algorithm, where the run realizes specifications formulated by the algorithm.
- ***specifiedBy***: A relation between an entity and the information content entity that specifies it.

2.3. Related ontologies

The following related ML ontologies are those that MLS is aligned to the moment. These alignments will be further described in the Section 3.

2.3.1. The OntoDM-core ontology

For the domain of data mining there are several developed ontologies, with the aim of providing formal descriptions of domain entities. One of the proposed ontologies is the OntoDM-core ontology. In one of the preliminary versions of the ontology, the authors decided to align the proposed ontology with the Ontology of Biomedical Investigations (OBI) [13] and consequently with the Basic Formal Ontology (BFO) at the top level[2], in terms of top-level classes and the set of relations. That was beneficial for structuring the domain in a more elegant way and the basic differentiation of information entities, implementation entities and processual entities. In this context, the authors proposed a horizontal description structure that includes three layers: a specification layer, an implementation layer, and an application layer. The specification layer in general contains information entities. In the domain of data mining, example classes are data mining task and data mining algorithm. The implementation layer in general contains qualities and entities that are realized in a process, such as parameters and implementations of algorithms. The application layer contains processual classes, such as the execution of the data mining algorithm.

2.3.2. The Exposé ontology

The main goal of Exposé is to describe (and reason about) machine learning experiments. It is built on top of OntoDM, as well as top-level ontologies from bioinformatics. It is currently used in OpenML, as a way to structure data (e.g. database design) and share data (APIs). MLS will be used to export all OpenML data as linked open data (in RDF).

[2] http://basic-formal-ontology.org/

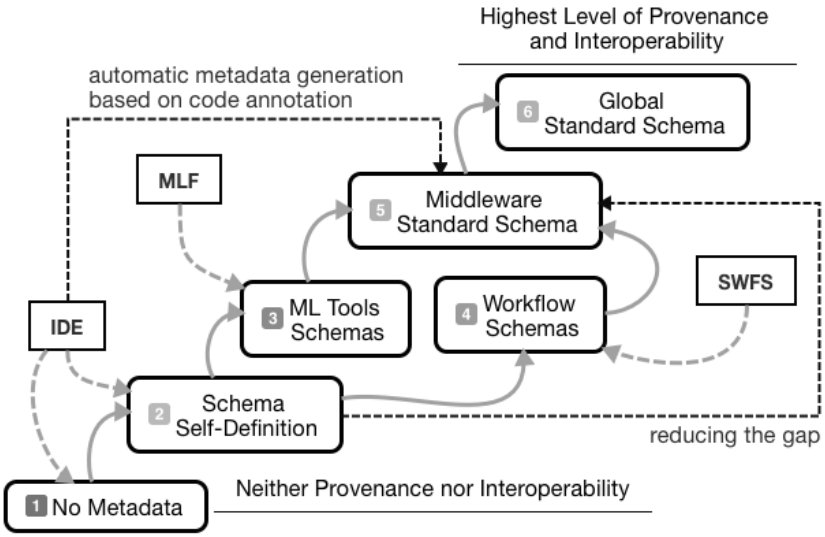

Fig. 1. Vertical and Horizontal Interoperability across ML Environments.

For the sake of simplicity and comprehension, we further refer to the Exposé ontology as the OpenML vocabulary, or simply OpenML.

2.3.3. The DMOP ontology

The DMOP ontology has been developed with a primary use case in meta-mining, that is meta-learning extended to an analysis of full DM processes. At the level of both single algorithms and more complex workflows, it follows a very similar modeling pattern as described in the MLS. To support meta-mining, DMOP contains a taxonomy of algorithms used in DM processes which are described in detail in terms of their underlying assumptions, cost functions, optimization strategies, generated models or pattern sets, and other properties. Such a "glass box" approach which makes explicit internal algorithm characteristics allows meta-learners using DMOP to generalize over algorithms and their properties, including those algorithms which were not used for training meta-learners.

2.3.4. The MEX vocabulary

MEX has been designed to reuse existing ontologies (i.e., PROV-O, Dublin-Core[3], and DOAP[4]) for representing basic machine learning information. The aim is not to describe a complete data-mining process, which can be modeled by more complex and semantically refined structures. Instead, MEX is designed to provide a simple and lightweight vocabulary for exchanging machine learning metadata in order to achieve a high level of interoperability as well as supporting data management for ML outcomes.

[3]http://dublincore.org
[4]http://usefulinc.com/doap/

3. MLS core and alignments

MLS provides a model for expressing data mining and machine learning algorithms, datasets, and experiments. This section introduces the related ontologies, the core of the MLS model – namely the classes (types) that are used to represent the majority of the cases – and the mappings with the existing ML ontologies. This mapping highlights how MLS is compatible with prior ontologies and how resources currently described in other ontologies can be described uniformly using MLS, hence allowing us to link currently detached machine learning resources.

*3.1. **Task***

In MLS, the Task class represents a formal description of a process that needs to be completed (e.g.based on inputs and outputs). A Task is any piece of work that needs to be addressed in the data mining process. Table 1 depicts a synthesis of the alignments detailed below.

3.1.1. OpenML

OpenML differentiates a TaskType (e.g. classification, regression, clustering,...) and Task instances.

The TaskType defines which types of inputs are given (e.g. a dataset, train-test splits, optimization measures) and which outputs are expected (e.g. a model, predictions,...). On the other hand, a Task contains specific dataset, splits, etc. It can be seen as an individual of the class.

3.1.2. DMOP

In DMOP, a task is any piece of work that is undertaken or attempted. A DM-Task is any task that needs to be addressed in the data mining process. DMOP's DM-Task hierarchy models all the major task classes: CoreDM-Task, DataProcessingTask, HypothesisApplicationTask, HypothesisEvaluationTask, HypothesisProcessingTask, InductionTask, ModelingTask, DescriptiveModelingTask, PredictiveModelingTask, and PatternDiscoveryTask.

3.1.3. OntoDM

OntoDM defines a data mining task as an objective specification that specifies the objective that a data mining algorithm needs to achieve when executed on a dataset to produce as output a generalization. It is represented as a subclass of the IAO: objective specification class, where objective specification is a directive information entity that describes and intended process endpoint. The data mining task is directly dependent of the datatypes of the data examples on which the task is defined, and is included directly in the task representations. This allows us to represent tasks defined on arbitrarily complex datatypes. The definition of data mining algorithm and generalizations is strongly dependent on the task definition.

OntoDM contains a taxonomy of data mining tasks. At the first level, we differentiate between four major task classes: predictive modelling task, pattern discovery task, clustering task, and probability distribution estimation task. Predictive modelling task is worked out in more detail. Since, a predictive modeling task is defined on a pair of datatypes (one describing the part of the data example on the descriptive side and the other describing the part of the data example on the target/output side), we differentiate between primitive output prediction tasks (that include among others the traditional ML tasks such as classification and regression) and structured output prediction tasks (that include among others tasks such as multi-label classification, multi-target prediction, hierarchical multi-label classification).

3.1.4. MEX

MEX has a higher level of abstraction, designed for representing ML executions and related metadata and not DM tasks. There are specific classes for representing specific ML standards. This information could be obtained from Learning Problem + Learning Method + Algorithm Class in a more concise level.

- Learning Problem: Association, Classification, Clustering, Metaheuristic, Regression, Summarization, ...
- Learning Method: Supervised Learning, Unsupervised Learning, Semi-supervised Learning, Reinforcement Learning, ...
- Algorithm Class: ANN, ILP, Bagging, Bayes Theory, Boosting, Clustering, Decision Trees, Genetic Algorithms, Logical Representations, Regression Functions, Rules, Support Vector Networks, ...

As an `:ExperimentConfiguration` may have many `:Executions` and an `:Experiment` may have many `:ExperimentConfigurations`, these can be aligned to a `mls:Task`.

Table 1
The syntheses of the MLS Task class and its relation with aligned ontologies.

Property	Value
Example Classes	Classification, Regression, Clustering, Feature Selection, Missing value imputation,...
Example Individuals	Classification on Dataset Iris
OpenML	TaskType
DMOP	DM-Task
OntoDM	"Data Mining Task"
MEX	The closest concept is `mexcore:ExperimentConfiguration`

*3.2. **Algorithm***

In MLS, the Algorithm class represents an algorithm regardless of its software implementation. Table 2 summarizes the alignments described below.

3.2.1. OpenML

OpenML currently does not abstract over algorithms anymore, it simply has 'implementations'. The underlying reasoning is that algorithms can come in endless variations, including hybrids that combine multiple pre-existing algorithms. Classifying every imple-

mentation as a specific type of algorithm is therefore not trivial and hard to maintain. Instead, to organize implementations, OpenML has 'tags', so that anybody can tag algorithms with certain keywords, including the type of algorithm that is implemented. Hence, hybrid algorithm can have multiple tags.

3.2.2. DMOP

An DM-Algorithm is a well-defined sequence of steps that specifies how to solve a problem or perform a task. It typically accepts an input and produces an output. A DM-Algorithm is an algorithm that has been designed to perform any of the DM tasks, such as feature selection, missing value imputation, modeling, and induction. The higher-level classes of the DM-Algorithm hierarchy correspond to DM-Task types. Immediately below are broad algorithm families or what data miners more commonly call paradigms or approaches. The Algorithm hierarchy bottoms out in individual algorithms such as CART, Lasso or ReliefF. A particular case of a DM-Algorithm is a Modeling (or Learning) algorithm, which is a well-defined procedure that takes data as input and produces output in the form of models or patterns.

3.2.3. OntoDM

In OntoDM, authors differentiate between three aspects of algorithms: algorithm as a specification, algorithm as an implementation, and the process of executing an algorithm. Data mining algorithm (as a specification) is represented as a subclass of IAO: algorithm. In this sense, a data mining algorithm is defined as an algorithm that solves a data mining task and as a results outputs a generalization and is usually published/described in some document (journal/conference/workshop publication or a technical report).

In OntoDM, it is given a higher level taxonomy of algorithms. At the first level, it is differentiated between single generalization algorithms (algorithms that produces a single generalization as a result) and ensemble algorithms (algorithms that produce an ensemble of generalizations as a result). At the second level, the taxonomy follows the taxonomy of tasks. This modular and generic approach allows easy extensions to characterize each algorithm class with its own distinctive set of characteristics that can be represented as qualities.

3.2.4. MEX

Sharing the problem stated by OpenML, MEX labels high levels of ML algorithms in Algorithm class instead of specific algorithm characterisations. As much as more precise information is needed, related classes could be instantiated, such as Learning Problem + Learning Method + Algorithm Class + Implementation.

Table 2
MLS Algorithm class alignments

Property	Value
Example Classes	Algorithm
Example Individuals	Linear Regression, Random Forest, AdaBoost...
OpenML	None
DMOP	DM-Algorithm
OntoDM	"Data Mining Algorithm"
MEX	mexalgo:Algorithm

3.3. *Implementation*

In MLS, the Implementation class represents an executable implementation of a machine learning algorithm, script, or workflow. It is versioned, and sometimes belongs to a library (e.g. WEKA). The alignments of this class against related ML ontologies are described following, and Table 3 summarizes them.

3.3.1. OpenML

OpenML does not distinguish between 'operators' and 'workflows', because the line is often very blurry. Many algorithms have complex internal workflows to preprocess the input data and make them more robust. Also, many environments (e.g. R, Matlab, etc.) do not have the concept of operator; they just have function calls, which are part of scripts. Hence, in OpenML, every implementation is called a Flow, which can be either atomic or composite.

3.3.2. OntoDM

In OntoDM, authors represent a data mining algorithm implementation as a subclass of OBI: plan is a concretization of a data mining algorithm. Data mining algorithms have as qualities parameters that are described by a parameter specification. A parameter is a quality of an algorithm implementation, and it refers the data provided as input to the algorithm implementation that influences the flow of the execution of algorithm realized by a data mining operator that has information about the specific parameter setting used in the execution process.

3.3.3. MEX

Implementation in MEX is meant to represent the *Software* Implementation and has no link to the algorithm itself. Examples are Weka, SPSS, Octave, DL-Learner.

Table 3
A syntheses of the MLS Implementation class and its alignments.

Property	Value
Example Classes	LearnerImplementation,DataProcessingImplementation,EvaluationProcedureImplementation
Example Individuals	`SVMlib,weka.J48,rapidminer.RandomForest,weka.evaluation.CrossValidation,weka.attributeSelection.GainRatioAttributeEval`
OpenML	Flow / Implementation
DMOP	DM-Operator / DM-Workflow
OntoDM	"Data mining algorithm implementation"
MEX	`mexalgo:Implementation`

3.4. *HyperParameter*

The MLS HyperParameter class represents a a prior parameter of an implementation, i.e., a parameter which is set before its execution (e.g. C, the complexity parameter, in `weka.SMO`). As shown in Table 4, this is directly aligned with OpenML's Parameter, MEX's AlgorithmParameter, and DMOP's OperatorParameter.

In OntoDM, however, a data mining algorithm execution is a subclass of SWO:information processing, which is an OBI:planned process. Planned processes realize a plan which is a concretization of a plan specification. A data mining algorithm execution realizes (executes) a data mining operator, has as input a dataset, has as output a generalization, has as agent a computer, and achieves as a planned objective a data mining task.

Data mining operator is a role of a data mining algorithm implementation that is realized (executed) by a data mining algorithm execution process. The data mining operator has information about the specific parameter setting of the algorithm, in the context of the realization of the operator in the process of execution. The parameter setting is an information entity which is a quality specification of a parameter.

Table 4
MLS HyperParameter class and its alignments

Property	Value
Example Classes	HyperParameter
Example Individuals	`weka.SMO_C,weka.J48_M,rapidminer.RandomForest_number_of_trees`
OpenML	Parameter
DMOP	OperatorParameter
OntoDM	Parameter
MEX	`mexalgo:AlgorithmParameter (mexalgo:HyperParameter` under proposal)

3.5. *Data*

In MLS, the Data class represents a data item composed of data examples and it may be of a various level of granularity and complexity. With regard to granularity, it can be a whole dataset (for instance, one main table and possibly other tables), or only a single table, or only a feature (e.g., a column of a table), or only an instance (e.g., row of a table), or a single feature-value pair. With regards to complexity, data examples are characterized by their datatype, which may be arbitrarily complex (e.g., instead of a table it can be an arbitrary graph). OpenML describes data at this level of granularity, while the alignment is more complex in other. Table 5 summarizes the alignments.

3.5.1. DMOP

DM-Data: In SUMO, Data is defined as an item of factual information derived from measurement or research. In IAO, Data is an alternative term for 'data item': 'an information content entity that is intended to be a truthful statement about something (modulo, e.g., measurement precision or other systematic errors) and is constructed/acquired by a method which reliably tends to produce (approximately) truthful statements'. In the context of DMOP, DM-Data is the generic term that encloses different levels of granularity: data can be a whole dataset (one main table and possibly other tables), or only a table, or only a feature (column of a table), or only an instance (row of a table), or even a single feature-value pair.

3.5.2. OntoDM

OntoDM imports the IAO class dataset (defined as 'a data item that is an aggregate of other data items of the same type that have something in common') and extends it by further specifying that a DM dataset

has part data examples. OntoDM-core also defines the class dataset specification to enable characterization of different dataset classes. It specifies the type of the dataset based on the type of data it contains. In OntoDM, we model the data characteristics with a data specification entity that describes the datatype of the underlying data examples. For this purpose, we import the mechanism for representing arbitrarily complex datatypes from the OntoDT ontology. Using data specifications and the taxonomy of datatypes from the OntoDT ontology, in OntoDM-core have a taxonomy of datasets.

3.5.3. MEX

In MEX, it is possible to represent even each instance (mexcore:Example) and each feature (mexcore:Feature) of the dataset.

Table 5
MLS Data class and its alignments.

Property	Value
Example Classes	Dataset, Train-test splits, Predictions
Example Individuals	Iris, FaceScrub, IMDB-WIKI
OpenML	Data
DMOP	DM-Data
OntoDM	Dataset specification, DM-dataset
MEX	`mexcore:Dataset` (as metadata)

*3.6. **Model***

We define Model as a generalization of a set of training data able to predict values for unseen instances. It is an output from an execution of a data mining algorithm implementation. Models have a dual nature: they can be treated as data structures and as such represented, stored and manipulated; on the other hand, they act as functions and are executed, taking as input data examples and giving as output the result of applying the function to a data example. Models can also be divided into global or local ones. A global model has global coverage of a data set, i.e., it generalizes the whole data set. A local model, such as a pattern set, is a set of local hypotheses, i.e. each applies to a limited region of the data set.

Table 3.6.2 demonstrates the alignments of MLS Model class that are described in the following.

Table 6
MLS Model class and its alignments

Property	Value
Example Classes	Decision tree, Rule set, Clusterings, Pattern set, Bayesian Network, Neural Net, Probability Distribution,...
Example Individuals	Decision tree built on Iris
OpenML	None
DMOP	DM-Hypothesis (with main subclasses: DM-Model, DM-PatternSet)
OntoDM	Generalization
MEX	None

3.6.1. DMOP

By Hypothesis, DMOP actually meant roughly ML models. They introduced the concept of a 'hypothesis' to differentiate ML models from pattern sets. On the other hand, the DM-PatternSet represents a pattern set, as opposed to a model which by definition has global coverage, is a set of local hypotheses, i.e. each applies to a limited region of the sample space.

3.6.2. OntoDM

In OntoDM, authors take generalization to denote the outcome of a data mining task. They consider and model three different aspects of generalizations: the specification of a generalization, a generalization as a realizable entity, and the process of executing a generalization.

Generalizations have a dual nature. They can be treated as data structures and as such represented, stored and manipulated. On the other hand, they act as functions and are executed, taking as input data examples and giving as output the result of applying the function to a data example. In OntoDM, a generalization is defined as a sub-class of the BFO class realizable entity. It is an output from a data mining algorithm execution.

The dual nature of generalizations in OntoDM is represented with two classes that belong to two different description layers: generalization representation, which is a sub-class of information content entity and belongs to the specification layer, and generalization execution, which is a subclass of planned process and belongs to the application layer.

In addition, a MLS ModelCharicteristic is a Generalization quality, while a MLS ModelEvaluation is mapped to a Generalization evaluation.

3.7. *Run*

An MLS run is an execution of an implementation on a machine (computer). It is limited in time (has a start and end point), can be successful or failed. If successful, it often has a specific result, such as a model and evaluations of that model's performance. Although runs are called very differently in the different existing ontologies, the semantics are the same. Table 3.7 shows the alignments with them.

Table 7
MLS Run class and its alignments

Property	Value
Example Classes	SimpleProcess, Execution
Example Individuals	Process running SVMlib on Iris on Machine m on timestamp *t*
OpenML	Run
DMOP	DM-Process (i.e., execution)
OntoDM	Data mining algorithm execution
MEX	`mexcore:Execution` (singly `mexcore:SingleExecution`, collectively `mexcore:OverallExecution`)

3.8. *EvaluationMeasure*

An MLS evaluation measure unique defines how to evaluate the performance of a model after it has been trained in a specific run. As shown in table 8, this is directly aligned across the different existing ontologies. In DMOP, however, there exist subclasses, such as ComputationalComplexityMeasure, HypothesisEvaluationMeasure, and ModelComplexityMeasure.

3.9. *Study*

An MLS study is a collection of runs that belong together to perform some kind of analysis on its results. This analysis can be general or very specific (e.g. an hypothesis test). It can also be linked to files, data, that belong to it. Studies are often the most natural product of a scientific investigation, and can be directly linked to certain claims and other products, such as research papers. As shown in Table 9, existing ontologies call this either a study or an experiment, although the semantics are the same.

Table 8
MLS EvaluationMeasure class and its alignments

Property	Value
Example Classes	`ClassificationMeasure`, `RegressionMeasure`, `ClusteringMeasure`, `RuntimeMeasure...`
Example Individuals	`Predictive_accuracy`, `root_mean_squared_error`, `inter_cluster_variance`, `cputime_training_milliseconds`
OpenML	`EvaluationMeasure`
DMOP	Measure
OntoDM	None
MEX	`mexperf:PerformanceMeasure`

Table 9
MLS Study class and its alignments

Property	Value
Example Classes	`BenchmarkStudy`
Example Individuals	Specific collections of runs
OpenML	Study
DMOP	DM-Experiment (i.e., something that resembles a bundle in PROV, e.g. `prov:Bundle`)
OntoDM	None
MEX	`mexcore:Experiment`

4. Use cases

To elucidate the benefits of MLS, we present three use cases where MLS can be utilized to foster the reproducibility of experiments. In particular, we show how previous research can benefit from the existence of an upper ontology which interlinks several vocabularies used for the exchange of experiment data and metadata.

4.1. *Open Provenance Model for Workflows and Research Objects*

It is often crucial to know exactly which data was used to train a machine learning model, where this data came from, and how it was processed before modelling. MLS is compatible with the *Open Provenance Model for Workflows (OPMW)* [14] and *Research Objects* [11]. This allows machine learning experiments to be described in a uniform way that preserves the provenance of data and models.

The term *provenance*, in computer science and scientific research, means metadata about the origin,

Table 10
Final full comparison between the terms of ML-Schema and aligned vocabularies

ML-Schema	OntoDM-core	DMOP	OpenML/Exposé	MEX Vocabulary
Task	Data mining task	DM-Task	Task	`mexcore:ExperimentConfiguration`
Algorithm	Data mining algorithm	DM-Algorithm	Algorithm	`mexalgo:Algorithm`
Software	Data mining software	DM-Software	N/A	`mexalgo:Tool`
Implementation	Data mining algorithm implementation	DM-Operator Algorithm implementation	N/A	`mexalgo:Implementation`
HyperParameter	Parameter	Parameter	Parameter	`mexalgo:HyperParameter`
HyperParameterSetting	Parameter setting	OpParameterSetting	Parameter setting	N/A
Study	Investigation	N/A	N/A	`mexcore:Experiment`
Experiment	N/A	DM-Experiment	Experiment	N/A
Run	Data mining algorithm execution	DM-Operation	Algorithm execution	`mexcore:Execution`
Data	Data item	DM-Data	N/A	`mexcore:Example`
Dataset	DM dataset	DataSet	Dataset	`mexcore:Dataset`
Feature	N/A	Feature	N/A	`mexcore:Feature`
DataCharacteristic	Data specification	DataCharacteristic	Dataset specification	N/A
DatasetCharacteristic	Dataset specification	DataSetCharacteristic	Data quality	N/A
FeatureCharacteristic	Feature specification	FeatureCharacteristic	N/A	N/A
Model	Generalization	DM-Hypothesis (DM-Model / DM-PatternSet)	Model	`mexcore:Model`
ModelCharacteristic	Generalization quality	HypothesisCharacteristic	Model Structure, Parameter, ...	N/A
ModelEvaluation	Generalization evaluation	ModelPerformance	Evaluation	N/A
EvaluationMeasure	Evaluation datum	ModelEvaluationMeasure	Evaluation measure	`mexperf:PerformanceMeasure`
EvaluationProcedure	Evaluation algorithm	ModelEvaluationAlgorithm	Performance Estimation	N/A

derivation or history of data or thing. For instance, in biology or chemistry, we track steps of experimental processes to enable their reproduction. In computer science, we track the creation, editing and publication of data, including their reuse in further processes. The PROV data model for provenance was created, founded on previous efforts such as Open Provenance Model (OPM) [15], and later became recommended by W3C [16]. The PROV Ontology (PROV-O), also recommended by W3C [17], expresses the PROV Data Model using the OWL language. PROV-O provides a set of classes, properties, and restrictions that can be used to represent and exchange provenance information generated in various systems. The Open Provenance Model for Workflows (OPMW) is an ontology for describing workflow traces and their templates which extends PROV-O and the ontology P-plan designed to represent plans that guided the execution of processes [14]. Figure 2 presents the mapping of the MLS directly to OPMW and indirectly to PROV-O and P-plan.

Belhajjame et al. [11] proposed a suite of ontologies for preserving workflow-centric *Research Objects*. The ontologies use and extend existing widely used ontologies, including PROV-O. Especially, the two ontologies from the suite, the Workflow Description Ontology (wfdesc), used to describe the workflow specifications, and the Workflow Provenance Ontology (wfprov), used to describe the provenance traces obtained by executing workflows, follow a very similar conceptualization of workflows to that of OPMW and map to MLS.

4.2. OpenML

The OpenML platform contains millions of machine learning experiments, which were run using thousands of machine learning workflows on thousands of datasets. However, in themselves, these experiments

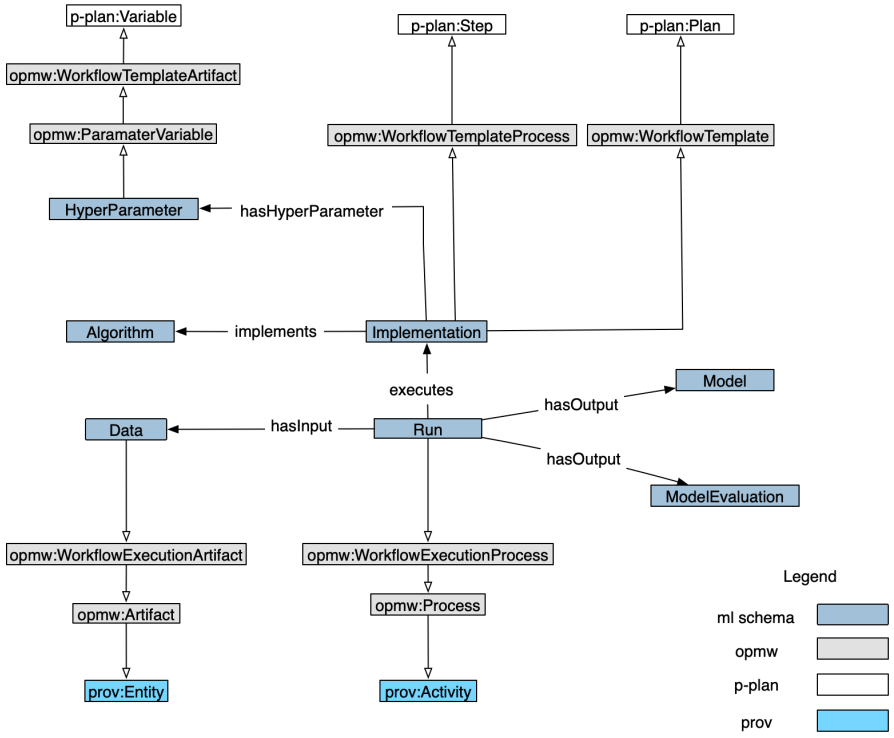

Fig. 2. The mapping of MLS to OPMW, PROV-O and P-plan.

form another island of data disconnected to the rest of the world. To remedy this, we have used MLS to describe all of these experiments as linked open data, so that scientists can connect their machine learning experiments to other knowledge sources, or build novel knowledge bases for machine learning research.

This is achieved through an export function that reads in OpenML's current JSON descriptions of datasets, tasks, workflows, and runs, and emits an RDF description using the MLS schema. This functionality is available as an open source Java library[5]. OpenML also supports this export functionality on the platform itself. In the web interface (openml.org) every dataset, task, workflow (flow), and run page has an RDF export button that returns the RDF description of that object, linked to other objects by their OpenML IDs. This functionality is also available via predictable URLs in the format https://www.openml.org/{type}/{id}/rdf, where *type* is either *d* (dataset), *t* (task), *f* (flow), or *r* (run), and *id* the OpenML ID of that object. Hence, the RDF description of dataset 2 can be obtained via https://www.openml.org/d/2/rdf.

As such, OpenML data becomes part of the semantic web, which allows scientists to link it to other data and reuse it in innovate new ways.

4.3. Deep Learning

This use case can also be described as a possible future work of MLS, where it is extended to support Deep Learning (DL) models.

By initiative of Microsoft and Facebook, a recently created community group called Open Neural Network Exchange (ONXX)[6] aims to allow users to share their Neural Network models and transfer them between frameworks. At the moment, it covers import/export to 3 different frameworks, while libraries for other 5 frameworks are under development or have partial support.

DL models have some requirements that MLS cannot describe at the moment – information such as number of layers and neurons, weights, and pre-trained

[5]The library is available on https://github.com/ML-Schema/openml-rdf

[6]https://onnx.ai/

models – as it only contains the HyperParameter class that is not able to store this additional information.

Unfortunately, the ONNX initiative does not provide an ontology; instead, their operators are described in the project GitHub documentation, while their terms are hardly defined in C code. On the other hand, the extension of the MLS ontology by adding new properties based on those terms would benefit not only the MLS, but all the aligned ontologies described in this work, that would instantly be able to use those properties to extend their models and support the description of DL models and experiments.

5. Conclusions and Future Work

In this paper we presented ML-Schema, a lightweight but sufficiently comprehensive easily extendable ontology for description of Machine Learning and support the description and open publication of such experiments in an interchangeable format. We demonstrated the extension of its expressiveness and how the MLS ontology was designed to be aligned with several ML ontologies, such as DMOP, OntoDM, MEX, and Exposé. It was also possible to elucidate through use cases the capabilities of our work, such as the usage of MLS format for exporting ML experiments to RDF format in the OpenML framework, its extension of that provides direct support to the OPMW and indirect to the PROV-O ontology, as well as the possible extension to elucidate the description of DL experiments. Such extension will be handled in the future discussions of the MLS Community Group, that welcomes everyone interested in extending our format to achieve a better support for description of ML experiments in an interchangeable format.

Acknowledgements

Gustavo Correa Publio acknowledges the support of the Smart Data Web BMWi project (GA01MD15010B) and CNPq Foundation (201808/2015-3). Agnieszka Ławrynowicz acknowledges the support from the National Science Centre, Poland, within the grant number 2014/13/D/ST6/02076. Pance Panov acknowledges the support of the Slovenian Research Agency within the grant J2-9230.

References

[1] Y. Gil, V. Ratnakar, J. Kim, P. Gonzalez-Calero, P. Groth, J. Moody and E. Deelman, Wings: Intelligent Workflow-Based Design of Computational Experiments, *IEEE Intelligent Systems* **26**(1) (2011), 62–72, ISSN 1541-1672. doi:10.1109/MIS.2010.9.

[2] O. Tcheremenskaia, R. Benigni, I. Nikolova, N. Jeliazkova, S.E. Escher, M. Batke, T. Baier, V. Poroikov, A. Lagunin, M. Rautenberg et al., OpenTox predictive toxicology framework: toxicological ontology and semantic media wiki-based OpenToxipedia, in: *Journal of biomedical semantics*, Vol. 3, BioMed Central, 2012, p. 7.

[3] D. De, R. Carole and G.R. Stevens, The design and realisation of the myexperiment virtual research environment for social sharing of workflows (2008).

[4] T.R. Gruber, A translation approach to portable ontology specifications, *KNOWLEDGE ACQUISITION* **5** (1993), 199–220.

[5] P. Panov, S. Džeroski and L. Soldatova, OntoDM: An ontology of data mining, in: *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, IEEE, 2008, pp. 752–760.

[6] C.M. Keet, A. Lawrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens and M. Hilario, The Data Mining OPtimization Ontology, *J. Web Sem.* **32** (2015), 43–53. doi:10.1016/j.websem.2015.01.001. https://doi.org/10.1016/j.websem.2015.01.001.

[7] J. Vanschoren and L. Soldatova, Exposé: An ontology for data mining experiments, in: *International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010)*, 2010, pp. 31–46.

[8] J. Vanschoren, J.N. Van Rijn, B. Bischl and L. Torgo, OpenML: networked science in machine learning, *ACM SIGKDD Explorations Newsletter* **15**(2) (2014), 49–60.

[9] D. Esteves, D. Moussallem, C.B. Neto, T. Soru, R. Usbeck, M. Ackermann and J. Lehmann, MEX vocabulary: a lightweight interchange format for machine learning experiments, in: *Proceedings of the 11th International Conference on Semantic Systems*, ACM, 2015, pp. 169–176.

[10] D. Esteves, A. Lawrynowicz, P. Panov, L. Soldatova, T. Soru and J. Vanschoren, ML Schema Core Specification, draft report, W3C Machine Learning Schema Community Group, 2016, http://www.w3.org/2016/10/mls/.

[11] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K.M. Hettne, R. Palma, E. Mina, Ó. Corcho, J.M. Gómez-Pérez, S. Bechhofer, G. Klyne and C.A. Goble, Using a suite of ontologies for preserving workflow-centric research objects, *J. Web Sem.* **32** (2015), 16–42. doi:10.1016/j.websem.2015.01.003. http://dx.doi.org/10.1016/j.websem.2015.01.003.

[12] C.B. Neto, D. Esteves, T. Soru, D. Moussallem, A. Valdestilhas and E. Marx, WASOTA: What Are the States Of The Art?, in: *SEMANTiCS (Posters, Demos, SuCCESS)*, 2016.

[13] R.R. Brinkman, M. Courtot, D. Derom, J.M. Fostel, Y. He, P. Lord, J. Malone, H. Parkinson, B. Peters, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, L.N. Soldatova, C.J. Stoeckert, J.A. Turner, J. Zheng and the OBI consortium, Modeling biomedical experimental processes with OBI, *Journal of Biomedical Semantics* **1**(1) (2010), 7, ISSN 2041-1480. doi:10.1186/2041-1480-1-S1-S7. https://doi.org/10.1186/2041-1480-1-S1-S7.

[14] D. Garijo and Y. Gil, Augmenting PROV with Plans in P-PLAN: Scientific Processes as Linked Data, in: *Second International Workshop on Linked Science: Tackling Big Data (LISC), held in conjunction with the International Semantic Web Conference (ISWC)*, Boston, MA, 2012. http://www.isi.edu/~gil/papers/garijo-gil-lisc12.pdf.

[15] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan and J.V. den Bussche, The Open Provenance Model Core Specification (V1.1), *Future Gener. Comput. Syst.* **27**(6) (2011), 743–756, ISSN 0167-739X. doi:10.1016/j.future.2010.07.005. http://dx.doi.org/10.1016/j.future.2010.07.005.

[16] P. Missier and L. Moreau, PROV-DM: The PROV Data Model, W3C Recommendation, W3C, 2013, http://www.w3.org/TR/2013/REC-prov-dm-20130430/.

[17] S. Sahoo, T. Lebo and D. McGuinness, PROV-O: The PROV Ontology, W3C Recommendation, W3C, 2013, http://www.w3.org/TR/2013/REC-prov-o-20130430/.

[18] H. Hanke and D. Knees, A phase-field damage model based on evolving microstructure, *Asymptotic Analysis* **101** (2017), 149–180.

[19] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J.M. Gómez-Pérez, S. Bechhofer et al., Using a suite of ontologies for preserving workflow-centric research objects, *Web Semantics: Science, Services and Agents on the World Wide Web* **32** (2015), 16–42.

[20] E. Lefever, A hybrid approach to domain-independent taxonomy learning, *Applied Ontology* **11**(3) (2016), 255–278.

[21] P.S. Meltzer, A. Kallioniemi and J.M. Trent, Chromosome alterations in human solid tumors, in: *The Genetic Basis of Human Cancer*, B. Vogelstein and K.W. Kinzler, eds, McGraw-Hill, New York, 2002, pp. 93–113.

[22] P.R. Murray, K.S. Rosenthal, G.S. Kobayashi and M.A. Pfaller, *Medical Microbiology*, 4th edn, Mosby, St. Louis, 2002.

[23] E. Wilson, Active vibration analysis of thin-walled beams, PhD thesis, University of Virginia, 1991.