LOG4MEX: A Library to Export Machine Learning Experiments

Diego Esteves
SDA Research Group
University of Bonn
Germany
esteves@cs.uni-bonn.de

Diego Moussallem
AKSW Research Group
University of Leipzig
Germany
moussallem@informatik.uni-leipzig.de

Tommaso Soru
AKSW Research Group
University of Leipzig
Germany
soru@informatik.uni-leipzig.de

Ciro Baron Neto
AKSW Research Group
University of Leipzig
Germany
baron@informatik.uni-leipzig.de

Jens Lehmann
SDA Research Group
University of Bonn
Germany
lehmann@cs.uni-bonn.de

Axel-Cyrille Ngonga Ngomo
University of Paderborn,
Data Science Group
Germany
axel.ngonga@upb.de

Julio Cesar Duarte
Military Institute of Engineering
Brazil
duarte@ime.br

ABSTRACT

A choice of the best computational solution for a particular task is increasingly reliant on experimentation. Even though experiments are often described through text, tables, and figures, their descriptions are often incomplete or confusing. Thus, researchers often have to perform lengthy web searches for reproducing and understanding the results. In order to minimize this gap, vocabularies and ontologies have been proposed for representing data mining and machine learning (ML) experiments. However, we still lack proper tools to export properly these metadata. To this end, we present an open-source library dubbed LOG4MEX which aims at supporting the scientific community to fulfill this gap.

KEYWORDS

LOG4MEX, Metadata, Machine Learning Experiments, Provenance, Ontology, Logging, Software Architecture

ACM Reference format:
Diego Esteves, Diego Moussallem, Tommaso Soru, Ciro Baron Neto, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, and Julio Cesar Duarte. 2017. LOG4MEX: A Library to Export Machine Learning Experiments. In *Proceedings of WI '17, Leipzig, Germany, August 23-26, 2017,* 7 pages.
https://doi.org/10.1145/3106426.3106530

WI '17, August 23-26, 2017, Leipzig, Germany

ACM ISBN 978-1-4503-4951-2/17/08…$15.00
https://doi.org/10.1145/3106426.3106530

1 INTRODUCTION

Nowadays, machine learning (ML) and data mining (DM) solutions have gained substantial attention due to a number of attractive solutions for existing scientific problems. However, researchers often take longer than expected to get a good understanding and process of all the information available in the solutions. This difficult interpretation of results happens because no shared standards exist to export and share experiment results implemented by current ML tools and frameworks[1]. A major consequence of this problem is the negative impact that a product created based on overestimated results of a given research activity can bring to society. A recent worrisome study showed that only 39 out of 100 replication attempts were successful. Furthermore, just 1 out of 61 experiments that could not be reproduced presented a "virtually identical" result [1] during the *Reliability test* [4]. Further analysis have found that only 6 of 53 high-profile papers in cancer biology could be reproduced [2]. One of the most famous examples of misleading research which we have to date is the *Potti scandal* at Duke University. This scandal is concerning the chemotherapy treatment, researchers needed about 2.000 hours just to find out the mistakes from the experimental setup and also to describe the negative impact of this on society [8]. To mitigate these risks, "publication", "code" and "data" should be available in repositories in order to enable *reproducible research* (RR) [11]. According to Peng, "*replication is the ultimate standard by which scientific claims are judged*". Reproducibility allows people to focus on the actual content of a data analysis, rather than on superficial details reported in a written summary. It is worth mentioning that in machine learning field, results are particularly hard to understand and/or replicate because of commonly missing technical details pertaining to the execution context and parameters of algorithms and tools[2].

[1] We do not consider *"workflow systems"*, once they are not mainly designed for machine learning. Instead, we focus on ML tools and libraries.
[2] https://www.wired.com/2017/04/want-fix-sciences-replication-crisis-replicate/

To this end, a ML [6] vocabulary and DM [9, 10, 13] ontologies have been proposed for creating well-defined structures based on common sense definitions. These schemata aim at defining standards to represent metadata derived from experiments, thus reducing the probability of misinterpretation, increasing the level of provenance, enabling interoperability and also reducing the human efforts for understanding the experiments. Furthermore, a recent group in W3C Community (ML-Schema Working Group)[3] has been started in order to combine efforts in this research area. Yet, generating this metadata is not a straightforward task and requires specialized knowledge for dealing with semantic web (SW) concepts and technologies, such as *Jena*[4].

For addressing these issues, we then designed a flexible and lightweight library called LOG4MEX[5]. LOG4MEX is based on MEX vocabulary [6][6], which is part of the MEX *open-source* project[7].

The main contributions of our work are as follows:

- We provide the first logging library for ML experiments based on state-of-the-art ML schemata.
- We define an architecture that allows extending the tool for supporting further ML vocabularies and ontologies.
- We show how semantic web technologies can minimize the effort of managing ML outcomes, allowing scientists to generate metadata without having to code extra scripts to export such configurations and outcomes.

The remainder of the paper is structured as follows: Section 2 presents real word examples of the problem related to interoperability, interpretation and data management. Section 3 briefly introduces the state-of-the-art ontologies. Section 4 details the architecture of LOG4MEX and shows the benefits of its usage. Section 5 progresses to discuss related works. Finally, Section 6 points out conclusions and possibilities for future works.

2 PROBLEM

In the following, we highlight problems which commonly occurs in ML experiments with regard to reproducible research. In order to make it easy and understandable, we analysed the available data[8] from one of the state of the art systems for Named Entity Recognition (NER) called FOX [12]. The format of the original data highlights the disadvantages of not having a standard structure to represent the experiment and its runs. Albeit, some assumptions can be made because of the good level of organisation in the physical folder structure, such as "*that might be the data for the first experiment*" and "*it should be the result for the token-based approach*". In the end, these assumptions are still thoughts. We then list the most relevant issues related to the models and parameters from FOX as follows:

- *Algorithm*: Originally "*Support Vector Machines (SVM)*" were invented by Vapnik in 1963 [14] and nowadays there are many different implementations and *kernel* derivations. Therefore, when the authors use the acronym *SVM*, it does not specify *a priori* the correct implementation.

- *Software and Parameters*: The authors cited the *Weka* framework [7] in order to define the values for default algorithm's parameters. But, this dependency may lead to misinterpretations once the values for the default parameters can either be changed or not declared by the tool's documentation. Also, the *software version* was omitted by authors, thus making the replication of this work even more difficult.

- *Model*: Foreseeing the Stanford NER *classifier* is also not suitable pertaining RR recommendations. For instance, depending which model (`eng.all.3class.distsim`, `eng.muc.7class.dist- sim` or `eng.conll.4class.distsim`) is performed on the following sentence "*A Hologram for the King - Tom Tykwer's latest movie hits German cinemas*") different results were achieved even just using 3 common classes.

- *Dataset*: Some datasets, such as "news" dataset, do not present a valid "landing page" which is responsible for directly connecting to the resource. Guessing here can also be tricky, leading the reader to different datasets containing similar names, such as "Yahoo News Dataset"[9].

The resolution of these issues are more detailed in Section 4.1 by using a more refined schema provided by LOG4MEX.

3 BACKGROUND

The MEX vocabulary [6][10] has been designed to tackle the problem of sharing provenance information particularly on the basic ML iterations in a lightweight format (i.e.: it aims at representing the flow *input* $\Rightarrow$ *model* $\Rightarrow$ *output* as simple as possible). The vocabulary is based on the PROV-O ontology [3] and consists of three main layers: (1) *mex-algo*, (2) *mex-core* and (3) *mex-perf* for representing *algorithms*, *executions* and its *outcomes*, respectively. (1) `mex-algo` represents the context of machine learning algorithms and their associated characteristics, such as learning methods, learning problem, hyper-parameters and its class. (2) `mex-core` formalizes the key entities for representing the basic steps on machine learning executions, as well as the provenance information for linking between the published paper and the produced meta-data. (3) `mex-perf` provides the basic entities for representing the experimental results of executions of machine learning algorithms. Figure 1 depicts an example of the most important entities of the vocabulary.

4 LOG4MEX

LOG4MEX is a library based on MEX vocabulary [6] which aims at reducing ML gaps by exporting ML outputs directly from source code independently of which ML library is used. The conceptual ML entities are mapped to its structure, making the metadata generation process easier to the end-user (once the process occurs in a transparent manner). The library complies with the software engineering best practices, thus producing an enriched meta-data file to share configurations of ML executions. LOG4MEX stands as a flexible and lightweight library to represent executions of algorithms and the related variables. Hence, LOG4MEX covers an important existing gap in standardization of ML approaches. Diverse areas which implement the flow $input(parameters) \Rightarrow algorithm(models) \Rightarrow outputs(measures)$ can benefit from the proposed library,

[3]https://www.w3.org/community/ml-schema/
[4]https://jena.apache.org/
[5]https://github.com/SmartDataAnalytics/mexproject/tree/master/log4mex
[6]https://github.com/SmartDataAnalytics/mexproject/tree/master/vocabulary
[7]https://w3id.org/mex
[8]https://github.com/AKSW/FOX/tree/master/evaluation
[9]https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=75
[10]https://github.com/SmartDataAnalytics/mexproject/tree/master/vocabulary

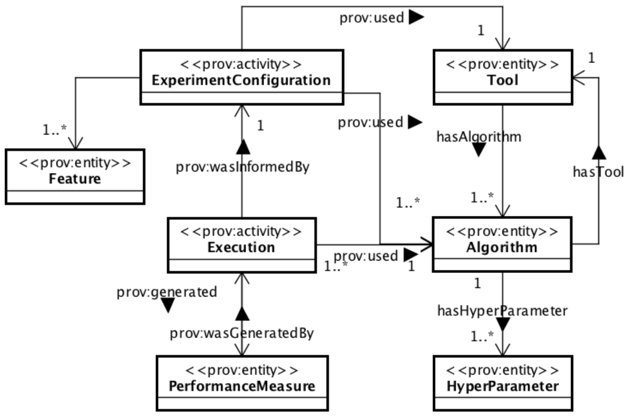

Figure 1: MEX Vocabulary: an example of the relationship among layers (*core*, *algo* and *perf*)

such as experiments in *natural language processing* or *stock market predictions*. A description of the architecture components depicted by the Figure 2 is available in Table 1. In addition, Table 2 describes the basic library's methods within common machine learning steps[11].

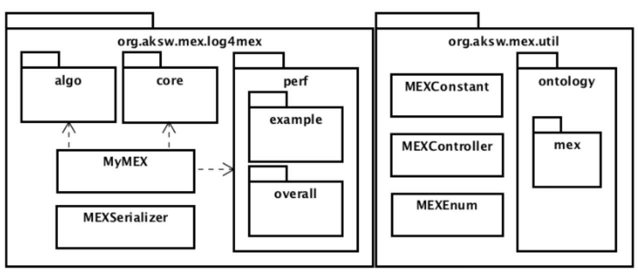

Figure 2: LOG4MEX component diagram: the modularization designed to keep the abstract concepts of machine learning. Furthermore, the package `ontology` has been designed to allow further Data Mining/Machine Learning schemata integrations.

Additionally, LOG4MEX is designed to encapsulate SW features (see Figure 2). For instance, the ontology package `org.aksw.mex.util.ontology` provides the mappings to upper-level ontologies, such as *PROV-O* and *DCAT*. Consequently, FAIR principles[12] are followed, collaborating with RR issues. The library[13] has two major classes: `MyMEX` and `MEXSerializer`. The first acts as a complex object to hold the ML variables as well as to add some authoring provenance information pertaining to the experimentation design. Whereas `MEXSerializer` is a *singleton* object responsible to parse and serialize the meta-data. **??** 1 shows an excerpt of code (`MyMEX` object) demonstrating its friendly interface and

[11] A full documentation can be found at the project website http://mex.aksw.org/.
[12] https://www.force11.org/group/fairgroup/fairprinciples
[13] Javadoc available at https://github.com/SmartDataAnalytics/mexproject/tree/master/documentation/log4mex

Table 1: LOG4MEX Architecture Components: MEX implementation.

Package (org.aksw.mex.log4mex*)	Description
*.algo	Mappings to mex-algo vocabulary
*.core	Mappings to mex-core vocabulary
*.perf	Mappings to mex-perf vocabulary
*.perf.example	Classes to represent performance of executions at *example* level [mexcore:SingleExecution]
*.perf.overall	Classes to represent performance of executions at *subset* level [mexcore:OverallExecution]
*.util	Static variables to map the vocabulary and control variables
*.util.ontology	Representation of diverse useful existing ontologies
*.util.ontology	Basic MEX classes types

Action	Description	Log4mex Method
Experiment Provenance	the basic experiment info	.setAuthorName(); .setAuthorEmail(); .setContext(); .setOrganization(); .setExperimentTitle(); .setExperimentDate(new Date());
Group of Executions	cluster the executions by logical sets - and define environment variables	.addConfiguration(); .Configuration().setDescription(""); .Configuration().setModel(); .Configuration().setHardwareConfiguration() .Configuration().setDataSet() .Configuration().setSamplingMethod() .Configuration().setTool() .Configuration().addFeature()
Algorithms and Hyperparameters	set algorithms and its parameters	.Configuration().addAlgorithm .Configuration().Algorithm().addParameter()
Executions	link executions and algorithms	.Configuration().Exec().SetAlgorithm()
Outcomes	get performance measures	.Configuration().Exec().addPerformance();

Table 2: LOG4MEX and standard machine learning processes

ease of use. The library integrated in ML scripts can more efficiently generate enriched metadata in *run-time*. Moreover, an RDF metadata file is easily generated with `MEXSerializer.getInstance().saveToDisk(<file path>, <uri>, obj, <format>, <ontology>);`.

```
/* library instance */
MyMEX mex = new MyMEX();
/* basic provenance */
mex.setAuthor("R. Speck", "speck@informatik.uni-
    leipzig.de");
mex.setContext(EnumContexts.NER);
mex.setOrganization("Leipzig University");
mex.setExperimentTitle("Ensemble Learning-NER");
mex.setExperimentDate(new Date('2015-08-04'));
mex.Configuration().setTool(WEKA, "3.6.6");
mex.Configuration().setHardwareConfiguration("
    ubuntu", EnumProcessors.INTEL_COREI7, EnumRAM.
    SIZE_16GB, "SSD", EnumCaches.CACHE_3MB);

```

```
/* algorithms and hyper−parameters */
alg1 = mex.Configuration().addAlgorithm(MEXEnum.EnumAlgorithm.NaiveBayes);
alg2 = mex.Configuration().addAlgorithm(MEXEnum.EnumAlgorithm.SVM);
alg2.addParameter("C", "1.0");
alg2.addParameter("E", "0.001");
/* features */
String[] features = {"min", "max", "ope clo"};
mex.Configuration().addFeature(features);

/* your model's call here !*/
...
/* mapping executions to algorithms */
String eid = mex.Configuration().addExecutionOverall(MEXEnum.EnumPhase.TEST);

/* assigning executions to outcomes */
mex.Configuration().ExecutionOverall(eid).setAlgorithm(alg1);
mex.Configuration().ExecutionOverall(eid).addPerformance(EnumMeasures.TruePositive, tp);
mex.Configuration().ExecutionOverall(eid).addPerformance(EnumMeasures.TrueNegative, tn);
mex.Configuration().ExecutionOverall(eid).addPerformance(EnumMeasures.FalsePositive, fp);
mex.Configuration().ExecutionOverall(eid).addPerformance(EnumMeasures.FalseNegative, fn);
mex.Configuration().ExecutionOverall(eid).addPerformance(EnumMeasures.F1measure, f1);
...
/* exporting metadata */
MEXSerializer.getInstance().saveToDisk(filename, URIbase, ml, MEXConstant.EnumRDFFormats.TTL);
```

Listing 1: A simple example of use for LOG4MEX. Basic authorship provenance information and two algorithms declared (*SVM* and *Naive Bayes*) for a specific group of executions (*Configuration*)

4.1 Application Scenario

Here, we show the benefits of using LOG4MEX. To this end, we present the solutions of the problems described in Section 2 regarding the work called FOX [12]. We then evaluate the approach[14] by comparing the current and produced output metadata. By using LOG4MEX we intend to guide and provide means to create a more robust representation of the existing variables and outcomes related to the experiment.

With respect to the introduced drawbacks, *algorithms* and their *hyperparameters* are set, respectively, by the methods `ml.Config().addAlgorithm(EnumAlgorithm x)` and `ml.Config().Algorithm(EnumAlgorithm x).addParameter(id,value)`.

[14]https://github.com/SmartDataAnalytics/mexproject/tree/master/examples/src/main/java/log4mex/fox

Analogously, `ml.Config().setTool(EnumTools y, version)` generates the link to the used version of the software and `ml.Config().setModel(name)` can be used to specify the classifier. Finally, the link to each dataset associated to each execution (or configuration) is set by `ml.Config().setDataSet(url, name)`.

In addition, LOG4MEX allows to achieve easily the interoperability among ML experiments over different system architectures (e.g.: Weka and JSAT). Therefore, any ML experiment which uses LOG4MEX, automatically abstracts ML concepts existing within the vocabulary (?? 2). The metadata is generated based in any of the existing RDF serialization formats.

```
this:m11 a prov:Entity, mexperf:ClassificationMeasure, mexperf:RegressionMeasure;
    dct:identifier "WekaPerformance";
    mexperf:accuracy "0.9768"^^xsd:float;
    mexperf:truePositive "147"^^xsd:integer;
    mexperf:falsePositive "3"^^xsd:integer;
    mexperf:kappaStatistics "0.97"^^xsd:float;
    ...
    prov:wasGeneratedBy this:ep1;.
```

Listing 2: An excerpt of the generated metadata file

Moreover, LOG4MEX handles the lack of automated and straightforward solutions for data management which requires to develop *wrappers* and implement *connectors* for a *Database Management Systems (DBMS)*, for instance. This extra step brings the focus out of the main problem being investigated, i.e., an extra code-effort is required to set up the desired environment. To this end, LOG4MEX has the advantage to generate automatically *RDF* data[15] (?? 3), thus no need to create scripts to generate the metadata is required.[16] Figure 3 illustrates the nature of generated metadata file which facilitates the navigation through entities of given experiment.

Figure 4 depicts a visual excerpt of code for the generated metadata file. Researchers are able to intuitively understand and absorb the relations of the variables in the experiment configuration. The visualization of an excerpt of metadata code generated by FOX use case: *dark blue* ellipses representing instance objects and *light blue* for classes. Gray ellipses are (incompletely) depicted in order to show further instances and their relations existing in the graph, providing a visual idea of the structure of connections.

Instead of trawling through the directory structure (more than 20.000 files in about 5GB of uncompressed data) and make suppositions, the generated metadata can be directly loaded into any triple store[17][18] and it can be queried by SPARQL queries (?? 4) and readily retrieved (Table 3).

```
SELECT DISTINCT ?executionID ?algorithm ?f1 WHERE {
  ?exec prov:id ?executionID.
  ?exec prov:used ?alg .
  ?p prov:wasGeneratedBy ?exec.
```

[15]https://www.w3.org/RDF/
[16]We do not intend to compare SQL and SPARQL approaches, but just highlight the disadvantage of using SQL Databases in this context, which require an extra code-effort as mentioned.
[17]https://www.w3.org/wiki/LargeTripleStores
[18]http://mex.aksw.org/sparql

Figure 3: An example of visualization of the produced graph: a rich metadata file linking the existing entities for a given experiment.

```
  ?p mexperf:f1Measure ?f1.
  ?alg a mexalgo:Algorithm.
  ?alg rdfs:label ?algorithm.
}
ORDER BY DESC (?f1)
LIMIT 4
```

Listing 4: One example of a generic SPARQL query for a *mex* output file: looking for the best top 4 configurations based on f1 scores

executionID	algorithm	f1
"C0_EXP1_EXEC_D44"	"BaggingJ48"	0.9968
"C0_EXP1_EXEC_D24"	"Logistic Model Trees"	0.9968
"C0_EXP1_EXEC_D16"	"Random Forest"	0.9968
"C0_EXP1_EXEC_D64"	"Multilayer Perceptron"	0.9967

Table 3: Output of (?? 4)

5 RELATED WORK

Recently, we have proposed an architecture [5] that guides the development of ML projects through *reflection*[19] and *annotation*[20] in order to automate a bit more the environment of coding. However, its architecture imposes constraints at the code level thus being a weakness. Therefore, to the best of our knowledge, LOG4MEX is the first library available to export ML metadata using SW technologies. LOG4MEX specializes the intention of LOG4J[21], but focusing on ML and built upon ontologies. Thus, allowing the generation of an enriched RDF metadata file instead of pure *text* files. Regarding to RR, there are others attempts to solve the problems described in Section 2, such as *Jupyter Notebooks*[22], which creates an interactive computational environment. It allows to create and share documents that contain live code, equations, visualizations and explanatory text. Also, different *workflow* systems and ontologies have been proposed to tackle this problem, providing an ecosystem either more complex (e.g: a framework for data mining tasks)

[19]https://en.wikipedia.org/wiki/Reflection_(computer_programming)
[20]https://en.wikipedia.org/wiki/Annotation
[21]http://logging.apache.org/log4j/2.x/
[22]urlhttps://ipython.org/notebook.html

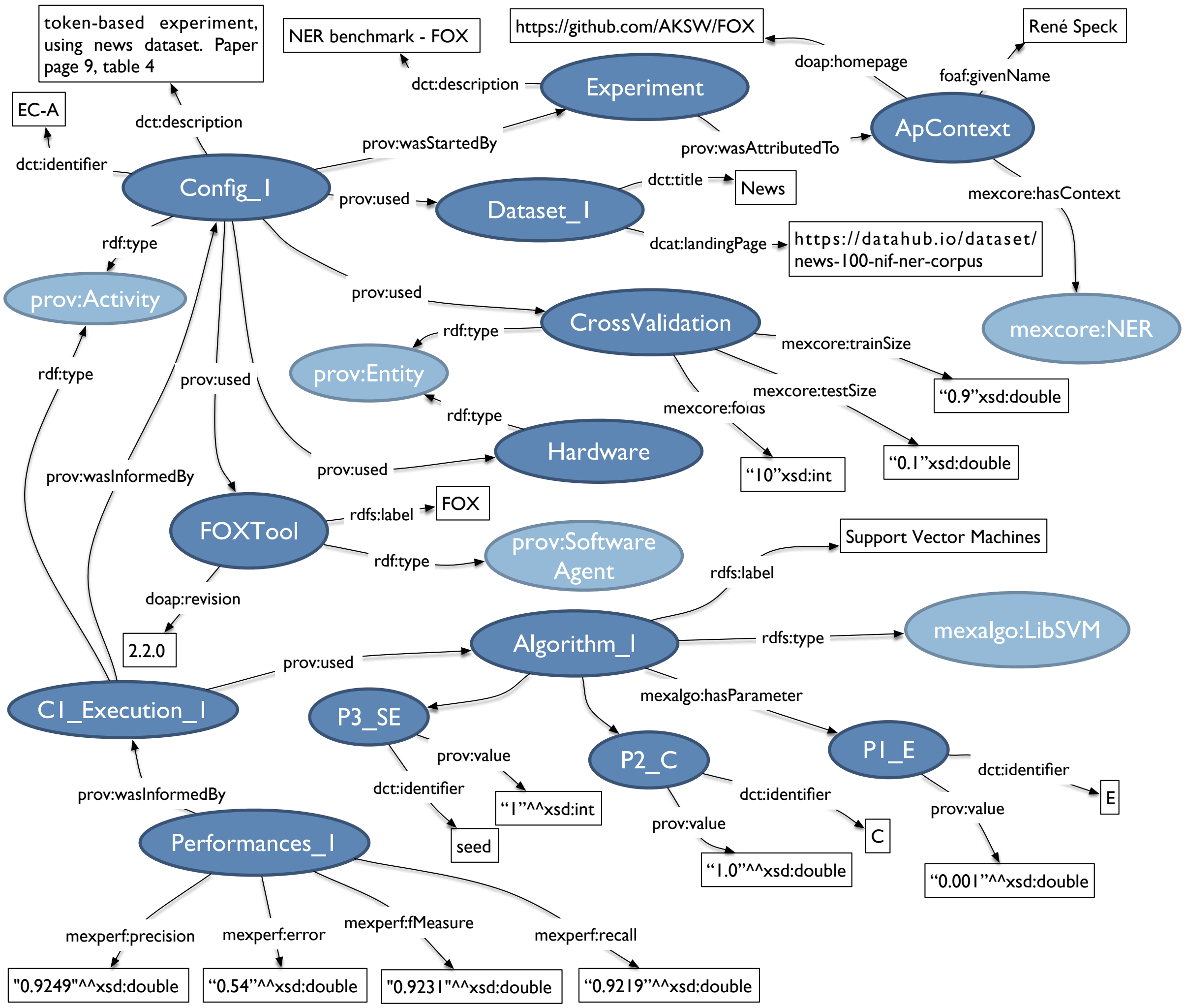

Figure 4: A more refined and semantically enriched metadata structure representing relations in the graph among instances.

or more specific to a certain task (e.g.: predictive toxicology). A comprehensive discussion has been introduced by Esteves et al., 2015 [6]. Thus, these approaches differ from LOG4MEX which proposes elements to export outcomes and parameters to RDF data directly from source-code associated to each machine learning algorithm execution. The main intention is to provide a logging library to export enriched machine readable metadata as simple as possible and with low dependency.

6 CONCLUSION

In this paper we discuss existing gaps at managing ML outcomes and its experiment parameters, regarding to RR issues. We also draw attention to the inevitable costs of development to both create interfaces and re-design schema to export the medatada. To bridge the gap, state-of-the-art vocabularies and ontologies have been proposed. However, the generation process of the proposed metadata is not straightforward, once specialized knowledge is required. In order to address this problem, we propose an architecture that encapsulates SW concepts and generate metadata in a straightforward manner. Hence, we released LOG4MEX, a library that aims to generate normalized metadata in a transparent process, facilitating the interoperability of results and data management. Finally, we analyzed the impact of the library in the context of an ensemble learning method for NER. Results showed the efficiency of our approach, minimizing the overall effort for seeking specific information as well as reducing the probability of misinterpretation. As future work, we plan stress the architecture in more complex machine learning scenarios, such as deep learning and also integrate the library into existing machine learning metadata portals, such as OpenML. Finally, we argue the metadata can be used to automatate the execution of machine learning pipelines based on its configurations.

```
@prefix this:<http://mex.aksw.org/examples/>.
@prefix mexcore:<http://mex.aksw.org/mex-core/>.
@prefix mexperf:<http://mex.aksw.org/mex-perf/>.
...
this:exp_cf_1_531468773_exe_1_phase
    a mexcore:Test ;
    rdfs:label "Test" .
this:exp_cf_1_531468773_exe_1_mea_1
    a mexperf:ClassificationMeasure ;
    rdfs:label "Classification measures" ;
    mexperf:accuracy "0.9646"^^xsd:double ;
    prov:wasGeneratedBy
        this:exp_cf_1_531468773_exe_1 .
this:exp_cf_1_531468773_exe_2
    a mexcore:ExecutionOverall ;
    rdfs:label "Overall Execution: C1_MEX_EXEC_D2"
        ;
    mexcore:endsAtPosition "section 18" ;
    mexcore:group true ;
    mexcore:startsAtPosition "section 0" ;
    mexcore:targetClass "Unknown words" ;
    prov:generated
        this:exp_cf_1_531468773_exe_2_mea_1 ;
    prov:id "C1_MEX_EXEC_D2" ;
    prov:used this:exp_cf_1_531468773_exe_2_phase
        , this:exp_cf_1_531468773_exe_2_algo ;
    prov:wasInformedBy this:exp_cf_1_531468773 .
...
```

Listing 3: An excerpt of RDF data produced by LOG4MEX

ACKNOWLEDGMENTS

This research has been partially supported by grants from the HOBBIT Project (GA no.688227), Big Data Europe (GA no. 644564) and the CAPES foundation, Ministry of Education of Brazil, Brasilia - DF 70040-020, Brazil (BEX 10179/13-5)

REFERENCES

[1] Monya Baker. 2015. First results from psychology's largest reproducibility test. Nature. (2015).

[2] C Glenn Begley and Lee M Ellis. 2012. Drug development: Raise standards for preclinical cancer research. *Nature* 483, 7391 (2012), 531–533.

[3] Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. 2012. *PROV-O: The PROV Ontology.* Technical Report. World Wide Web Consortium. http://www.w3.org/TR/prov-o/

[4] Open Science Collaboration and others. 2015. Estimating the reproducibility of psychological science. *Science* 349, 6251 (2015), aac4716.

[5] Diego Esteves, Pablo N. Mendes, Diego Moussallem, Julio Cesar Duarte, Amrapali Zaveri, and Jens Lehmann. 2016. MEX Interfaces: Automating Machine Learning Metadata Generation. In *Proceedings of the 12th International Conference on Semantic Systems (SEMANTiCS 2016).* ACM, New York, NY, USA, 17–24.

[6] Diego Esteves, Diego Moussallem, Ciro Baron Neto, Tommaso Soru, Ricardo Usbeck, Markus Ackermann, and Jens Lehmann. 2015. MEX Vocabulary: A Lightweight Interchange Format for Machine Learning Experiments. In *Proceedings of the 11th International Conference on Semantic Systems (SEMANTICS '15).* ACM, New York, NY, USA, 169–176.

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter* 11, 1 (2009), 10–18.

[8] Darrel Ince. 2011. The Duke University scandal. *Significance* 8, 3 (2011), 113–115.

[9] C. Maria Keet, Agnieszka Ławrynowicz, Claudia d'Amato, Alexandros Kalousis, Phong Nguyen, Raul Palma, Robert Stevens, and Melanie Hilario. 2015. The Data Mining OPtimization Ontology. *Web Semantics: Science, Services and Agents on the World Wide Web* 32 (2015), 43 – 53. DOI:https://doi.org/10.1016/j.websem.2015.01.001

[10] Panče Panov, Larisa Soldatova, and Sašo Džeroski. 2013. OntoDM-KDD: Ontology for Representing the Knowledge Discovery Process. In *Discovery Science.* Springer Berlin Heidelberg, Heidelberg, Berlin, 126–140. DOI:https://doi.org/10.1007/978-3-642-40897-7_9

[11] Roger D Peng. 2011. Reproducible research in computational science. *Science (New York, Ny)* 334, 6060 (2011), 1226.

[12] René Speck and Axel-Cyrille Ngonga Ngomo. 2014. Ensemble learning for named entity recognition. In *The Semantic Web–ISWC 2014.* Springer, Cham, 519–534.

[13] Joaquin Vanschoren and Larisa Soldatova. 2010. Exposé: An ontology for data mining experiments. In *International workshop on third generation data mining: Towards service-oriented knowledge discovery (SoKD-2010).* Springer, Ljubljana - Slovenia, 31–46.

[14] Vladimir Naumovich Vapnik and Vlamimir Vapnik. 1998. *Statistical learning theory.* Vol. 1. Wiley New York, New York, NY, USA.