Classifying Unstructured Models into Metamodels Using Multi Layer Perceptrons

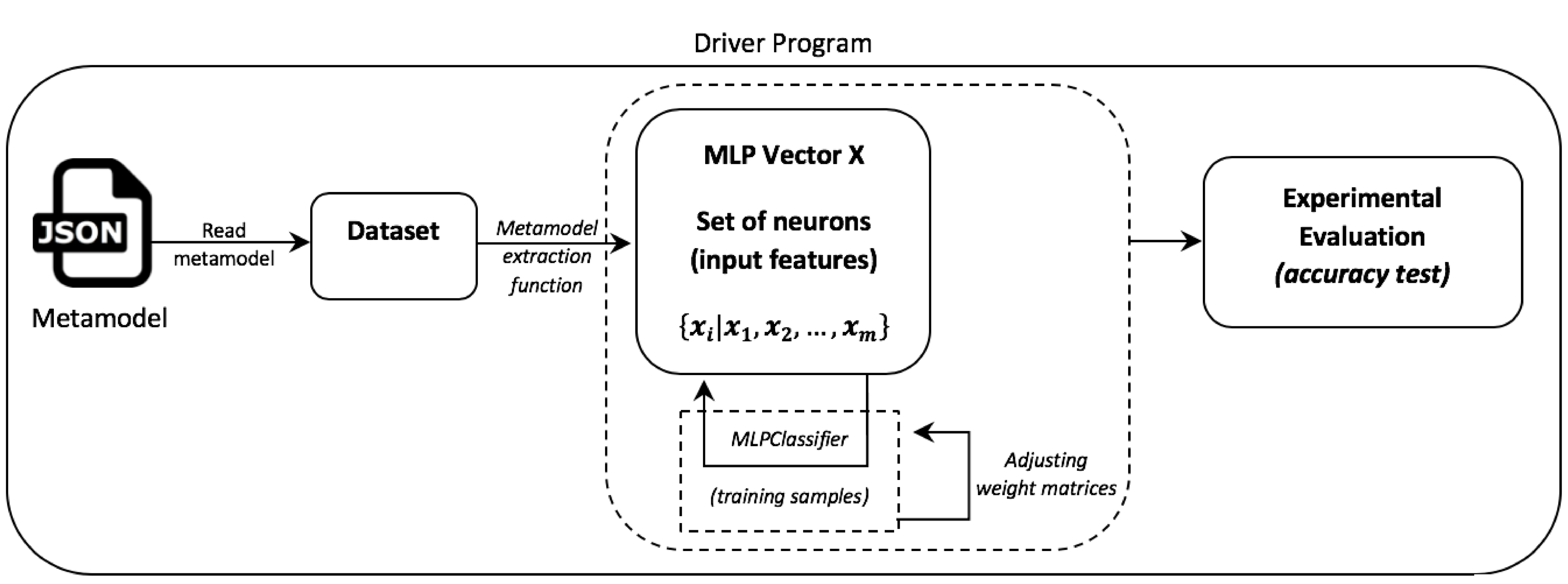
Walmir Oliveira Couto[1,2], Emerson Cordeiro Morais[2] and Marcos Didonet Del Fabro[1]

[1]C3SL Labs, Federal University of Paraná (UFPR), Curitiba PR, Brazil - [2]LADES Icibe, Federal Rural University of Amazon (UFRA), Belém PA, Brazil
{walmir.couto, emerson.morais}@ufra.edu.br, marcos.ddf@inf.ufpr.br –> Code can be found on github: https://github.com/walmircouto/MLPTraining

Introduction

JSON documents are used for interoperability, storage of application data where flexibility is important and also are becoming a de-facto standard in RESTful APIs implementations. When JSON schemas are not defined, it is useful to classify JSON documents to discover whether they could be categorized into a given domain and to partially-conform to a metamodel. Recent approaches are emerging to extract information or to couple metamodeling with cognification [1], some approaches use Long Short-Term Memory Neural Networks (LSTM) to automatically infer model transformations from sets of input-output model pairs [2], another one employ Machine Learning techniques for metamodel automated classification [3]. We present a methodology to analyze and classify JSON documents according to existing metamodels. We extract existing metamodels using a One-hot encoding solution into a Multi-Layer Perceptron (MLP) network, translating the metamodel elements into the input neurons. The neural network is used to classify input JSON documents, which are as well translated into the input data to be classified. We have conducted a series of experiments, using neural networks with different intermediate layers, showing that the approach is effective to classify the documents.

Classifying Document into Metamodels



The *Driver Program* implements the control flow and it launches the operations, managing the step by step execution schema:

- Reading JSON input metamodel and assigns it to a data set;
- Processing the data set using an *extraction* function, which selects classes, attributes and references;
- Once this conversion is done, there is no distinction between the types of the elements to encode the MLP features. Then, each one of these data set is converted into a binary number and bundled to create the *MLP Vector X: set of input layer neurons* $\{x_i|x_1, x_2, ..., x_m\}$ applying One-hot Encoding (OHE) technique because categorical data must be converted to numbers;
- Training the network by *MLP Classifier* using a set of training samples;
- There is no difference between classes, attributes, and references, each one is a binary number in *MLPVectorX*.

We define an algorithm which reads JSON input metamodel, applies an *extraction* function to take distinct classes, attributes and references amount which we use to calculate the binary digits amount used to depict these classes, attributes and references in a binary vector which it is used as input features on MLP.

***MLPVectorX* structure for Java metamodel**			
Element	**Type**	**Position**	**Value**
JavaElement	class	0	0000000
name	attribute	1	0000001
Type	class	2	0000010
Modifier	class	3	0000011
isPublic	attribute	4	0000100
isStatic	attribute	5	0000101

Results

Evaluating MLP with 5 Hidden Layers				
Models with 50 elements				
%	**MySQL**	**KM3**	**UML**	**Java**
100%	100%	100%	100%	100%
Models mixed				
%	**MySQL + KM3**		**UML + Java**	
80%-20%	97,2%		96,6%	
60%-40%	87,3%		85,6%	
50%-50%	48,6%		47,3%	
Models with 100 elements				
%	**MySQL + KM3**		**UML + Java**	
80%-20%	97,6%		95,1%	
60%-40%	83,8%		87,6%	
50%-50%	47,7%		46,8%	

Documents with elements 100% according to their respective metamodel, MLP correctly classifies all the documents. When the elements are mixed, MLP with 3 hidden layers showed 96,3% of precision, and the MLP with 5 hidden layers showed 97,2%, improving 0,9%. This means adding two layers had a small impact on the final result.

Conclusion

The solution enables discovering the domain of the JSON documents and to serve as an initial typing scheme. We present the automated steps of the approach, consisting on metamodel extraction into an MLP using a one-hot encoding (OHE) of the elements, network training, translation and classification of the input JSON documents. The results have showed that the approach is effective from classifying JSON documents, with precision varying from 46 to 97 percent, depending on the kinds of the elements.

References

1. Cabot, J., Clarisó, R., Brambilla, M., and Gérard, S. Cognifying model-driven software engineering (2017).
2. Burgueño, L. An lstm-based neural network archi-tecture for model transformations. MODELS (2019).
3. Nguyen, P., Di Rocco, J., Di Ruscio, D., Pierantonio,A., and Iovino, L. Automated classificationof metamodel repositories. MODELS (2019).