A Lightweight Framework for Uniform Experiments

Abstract

Lorenz Weizsäcker
Knowledge Engineering Group, Technische Universität Darmstadt

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Knowledge
Engineering

Abstract

In the area of experimental machine learning a wide range of frameworks have been developed and published that allow the experimenters to employ approved, high level tools for divers purposes. Roughly speaking, there are two types of framework incentives: the *engine* type wants to provide code that solves a specific type of task. In order to employ it the user is expected to build a framing code by which the provided engine is called. The *embracing* framework incentive wants that experiments are run through the framework itself which subsequently calls the engines that can be plugged-in in terms of wrappers.

Although writing the embracing experimental code usually is straightforward, it can be error-prune and time consuming and it therefore is plausible to share such code as well. However, defining an embracing framework is rather ambitious as it somehow has to formalize the intentions of potential users with respect to the experimental setup and the employed data structures. A trade-off between comfort versus generality is inevitable: a narrow specification on what experiments are allows broad pre-build services for the specified type of experiments but it also make the framework less likely to be applicable to the different projects of the users.

In the talk we want to present an attempt to build a novel embracing framework that sets the trade-off much more in favor of generality than many other machine learning frameworks do. It can offer much less services and completely forsakes the user in terms of productive pieces of code such as preprocessing or training engines. Also, it comes with its own restriction. But these restriction concern the interplay between the experimental components rather the type of learning task or the employed data structures. Since it is indifferent in such regard it can also be relevant for experimental research other than empirical machine learning.

In order to use the proposed system we first decompose the experiment into components that are called e-nodes. It is not properly specified what an e-node is but it somehow should represent a well-rounded concept within an experiment. Examples of e-nodes are datasets, certain preprocessing, specific part of the training such as building of a topic model, training parameters, or performance measures. The e-nodes represent the dimension of an experiment, and we assume that the experimenter wants to try out uniform combination of different values for at least some of those dimension. The most typical examples for such experimental dimensions are different dataset, and real valued training parameters. The trick of the framework is, however, that we entirely decompose the experiment into such dimensions whether we want to apply different values to them or not. The restriction of the framework is, roughly speaking, that each dimension has a fixed number of values.

If the user adapts the way of coding experiments to e-node decomposition, we can make the system understand to some extend the structure of the experiment. This enables modest but useful pre-build services as an increased persistence of names: the system demands the user to provide names for the dimension/e-nodes as well as for their values if the values themselves cannot serve as names. In return, it extends the lifetime of the names in the sense that they can be used for assembling the experiment, but also for querying and presentation of the results.

By now, we cannot soundly assess the potential utility of the framework, because we first have to apply its model of experiments to larger set of different settings. In any case the choice of frameworks is a matter of taste. The system we want to present in the talk particularly addresses users that are disposed to formalization but sometimes get confused by their own code and the output it produces. A prototype implementation of the framework can be downloaded at `http://www.ke.tu-darmstadt.de/resources/peewit`.