LLM installation instructions

Irma Ravkic
irma.ravkic@cs.kuleuven.be

1 Introduction

Learner of local models (LLM) is implemented in Java. The algorithm uses Prolog as a module for calculating feature values. The communication with Prolog is done by means of a 2p.jar located in the library. It is fully integrated and, therefore, there is no demand on users to set anything for this interface to work properly.

2 Prerequisite

The user needs Java Runtime Environment (JRE) 1.7 to run the code and Java Development Kit (JDK) 1.7 or newer to compile it. For compiling the code and generating executable jar, the user needs Apache Maven 3.0.5 or newer. For JRE, JDK and Apache Ant, the most stable versions for your platform are the best choice.

3 Instructions

The following installation instructions are valid for all platforms that support Java and Apache Maven. In order to generate needed executable jar, go to the main folder "HybridRDN". Then run the following command:

```
mvn clean compile assembly:single
```

This command will compile all the classes and generate an executable jar `llm.jar` in folder `target`.

4 Test

In order to test whether the executable jar is functional perform this little test in the terminal. All the needed files are provided in folder `UniversityExample /data_university`.

```
java -jar target/llm.jar -input
UniversityExample/data_university/ -output
UniversityExample/results/ -predicates intelligence
```

This command will run the llm learner using the main class of the default `hybrid.custom_structure_learner.Custom_StructureLearner`. When the program finishes the user should have `results` folder with results for `intelligence` predicate in `UniversityExample` folder. The information about these files is given in the manual provided in this folder as well. This runnable jar executes main method in `custom_structure_learner.Custom_StructureLearner`. This method represents the guideline on how to run our structure learning algorithm.

5 Where from here?

As mentioned before, the runnable jar file created above runs the main class of `hybrid.custom_structure_learner.Custom_StructureLearner` class. There are some important parts of this class that a user can change to run the algorithm for the network and data he/she wants:

1. `Network hybrid_university=new Network();`
   `ntw=hybrid_university.defineApplicationNetwork(1);`
   Here we create a network class that has random variable declarations (RVDs). In this default class we give RVDs of the University domain. In order to create one from scratch, the user should look into the `manual.pdf` where we provided detailed instructions.

2. `Data d_training=dataLoader.loadData(parameters.input_path+"/train/",`
   `"interp", "pl",ntw);`
   `Data d_validation=dataLoader.loadData(parameters.input_path+"/validate/",`
   `"interp", "pl",ntw);`
   `Data d_test=dataLoader_no_subsampling.loadData(parameters.input_path+`

`"/test/","interp", "pl",ntw);`
The user should specify the data used for his/her purposes.

The rest of the main class should stay more or less the same. When the classes are changed for the purposes of the user, a new executable jar should be made with the procedure explained above.