YINYANG: Release Notes

Luigi Iannone

May 31, 2006

1 Scope

This documents illustrates the basic features of YINYANG system. The authors as well as YINYANG developers are not responsible of any mistake in this document and/or within YINYANG system

2 Requisites and running instructions

YINYANG has been tested on Java 1.5 both on MAC OS 10.4.6 and on Windows XP/2000 systems. What you need is a JRE 1.5 and YINYANG binaries. YINYANG can be started from command line. In Fig. 1 is the typical UNIX-like command line (for Windows just substitute ':' with ';' in the classpath).

```
java -cp
yinyang.jar:rdf2kb.jar:abstraction.jar:kb.jar:learningProblem.jar:logger.jar:
dig1.1-reasoners.jar:dig1.1-xmlbeans.jar:jaxen-1.1-beta-2.jar:xbean.jar:
xbean_xpath.jar:
antlr-2.7.5.jar:commons-logging.jar:concurrent.jar:icu4j_3_4.jar:iri.jar:
jena.jar:json.jar:junit.jar:log4j-1.2.12.jar:xercesImpl.jar:xml-apis.jar:
it.uniba.di.dl.learningsystems.control.LearningController
<pathForTheLearningProblem> <pathForTheResultLogFile>
```

Figure 1: Command Line

It has two input parameters:

1. ¡pathForTheLearningProblem¿: the path where the learning problem file is (see Sect. 3 for further details on leanring problem specification)
2. ¡pathForTheResultLogFile¿: the path where to log the system debug information (optional)

3 Learning Problem

A learning problem for YINYANG is an XML file in the format in Fig. 2.

```
<?xml version="1.0" encoding="UTF-8"?><LearningProblem
xmlns="http://www.di.uniba.it/learning"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.di.uniba.it/learning ../LearningProblem.xsd"
knowledgeBase="file:./basicFamily.owl"
positiveExampleSource="./basicFamilyBrotherPositives.txt"
negativeExampleSource="./basicFamilyBrotherNegatives.txt"
abstractionClass="it.uniba.di.dl.abstraction.KBoundMSCApproximatorRefinement"
learn="true" normalize="false" consistencyCheck="true"> <Experiments/>
<Namespaces> <Namespace
uri="http://www.csc.liv.ac.uk/~luigi/ontologies/basicFamily#"></Namespace>
</Namespaces> <Output
learnedConceptURI="http://www.csc.liv.ac.uk/~luigi/ontologies/basicFamily#Brother"
outputPath="./LearnedBrother.owl" showOnConsole="true"></Output>
</LearningProblem>
```

Figure 2: Example of Learninf problem]

The Learning problem tag has the following attributes:

- knowledgeBase: The URL of the background knowledge that contains the descriptions of the individuals (it has to be a reachable OWL ontology)
- positiveExampleSource: the path of a file containing the list of positive examples URIs
- negativeExampleSource: the path of a file containing the list of negative examples URIs
- normalize: if true asks the system to normalize the learnt concept
- consistencyCheck: if true asks the system to perform consistency check on the training set (i.e.: checks whether the learnt concept actually covers all the positives and no negative)
- learn: if false skips learning and performs the experiments (if any)
- reasonerURL: (optional) determines the URL of the DIG reasoner HTTP service (default `http://localhost:8080`)

The Namespaces tag contains a list of Namespace children tags. Each of them indicates one of the namespaces that should be taken into account in the learning process (uri attribute). That means that if an OWL resource has a namespace not appearing in that list, it will be dismissed by the learner.

Experiments tag contains a list of possible experiments that can be performed. We have three kinds of experiments corresponding to the most common experimental settings (we enumerate them using the corresponding tags):

1. SeventyThirty: It divides the data set according to the optional precentage attribute (default is 70% for tranining and 30 % for validation)
2. kFoldCrossValidation: It divides the dataset in k folds (randomly) where k is determined by the folds attribute (optional - default 10). It uses k-1 folds for training and 1 for validation. It repeats the experiment rotating the validation set across until all the folder have been used once for validation and averages the results
3. LeaveOneOut: It uses just one randomly chosen example as validation and all the rest for training. it rotates the validation until all the examples have been used for validation. It averages the results.

Each of the above tags have (likewise Learning problem) the following attributes:

- positiveExampleSource: the path of a file containing the list of positive examples URIs
- negativeExampleSource: the path of a file containing the list of negative examples URIs

The Output tag (optional) has the following attributes

- learnedConceptURI: the URI the learned concept will have
- outputPath: The path of the knowledge base enriched with the new learned definition
- showOnConsole: if true shows the saved OWL enriched knowledge base on the console as well (optional - default true)

4 Acknowledgement

Luigi Iannone wishes to thank in particular Mimmo Redavid for his patience and his ideas that stimulated the development of this system.