RJEMM

Created by: Benjamin Koopferstock

About RJEMM

RJEMM is an extension of the Java implementation of Extensible Markov Models. Using the rJava package for R, RJEMM calls various methods in JEMM, so that JEMM can be used from within R, and has some added features using other R packages. RJEMM is very easy to use and is well documented within R. This document will mostly repeat what is already said in the R documentation pages. RJEMM was developed in the spring semester of 2009 by Benjamin Koopferstock in a special topics course with Dr. Dunham at Southern Methodist University.

Install RJEMM

Installation within R is very simple. Simply enter

> install.packages("*RJEMM*",repos="http://R-Forge.R-project.org")

within R, and the package should install automatically from the R-Forge page.

Creating a Model

Building a model in RJEMM is not a difficult task. Basic documentation is available through the R package, so these instructions will be a bit more detailed.

Once R has been started and the package installed, simply type

> library(RJEMM)

to open up the package for use. RJEMM comes with a variety of data sets for testing, in this example we will use the anomaly data set. To load the dataset, enter

> data(anomaly)

Now, to create an EMM object, use the create_EMM method. Using this method, the threshold and clustering method can be set. In this example, the Dice measure is used and the threshold is set to 0.5. Other possible measures are "Cosine", "Dice", "Euclidean", "Jaccard", and "Overlap". For the threshold, use a value between 0 and 1.

> emm <- create_EMM(measure = "Dice", threshold = 0.5)

Now that the data is loaded in R and the RJEMM object is created, it is time to build the model using the data. To do this, enter

> build(emm,anomaly)

and now your model is complete.

Examining the Model

RJEMM comes with several tools for inspecting an EMM. Two closely related methods for output are displayStates and displayLinks, which output the states and links between the states for an EMM. To call these methods, type

> displayStates(emm)

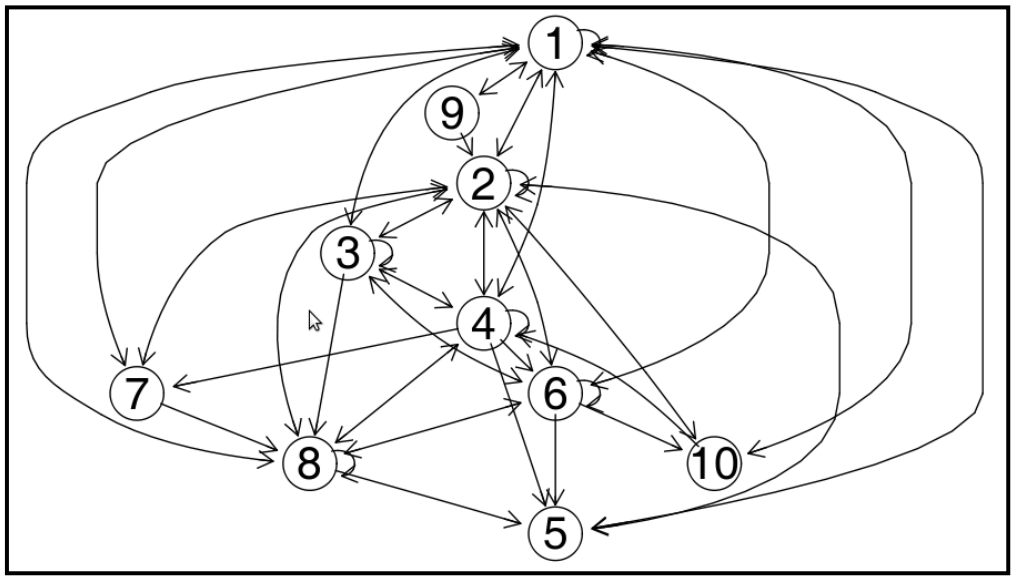
and

> displayLinks(emm)

Another useful way to examine the model is with the graph visualization tool. Type

> display(emm)

and a graph visual should pop up, displaying the states and links between the states. If you built an EMM using the data and parameters given under the “Creating a Model” heading, your graph should look like this:



Saving and Loading Models

Saving and Loading models is very easy to do. Models are saved and loaded in the same format as in JEMM, so models built in either application can be opened in the other. Saving a model is done through the save() method, which has just one parameter, the location to save. Calling

> save("/home/ben/Documents/EMM/EMMModel.bin", emm)

would save the emm model to the file EMMModel.bin in my /home/ben/Documents/EMM directory.

Loading a model is done using the create_EMM() method. If the location parameter in the create_EMM() method is set, then a model will be loaded. To load the above model, I would call

> emm <- create_EMM(location="/home/ben/Documents/EMM/EMMModel3.bin")

the model would then be loaded as an RJEMM object and any methods can be called on it.

Detect Rare Events

Once a model has been built, one can add new data to the model and display any rare events that occur in the new data using the detectEvent() method. After building an EMM based on the anomaly data, as shown above, the following code can be used to add new data and display rare events. An example data set is given with the package that can be used to demonstrate rare event detection. To load data from the example given, enter

> data(anomalyRare)

Now that the data has been loaded, the detectEvent() method can be used to add data to the model and print out any rare states. Calling

> detectEvent(emm, anomalyRare, threshold = 0.7, measure="Cosine")

will print out the rare states. The threshold and measure can also be set when adding new data, as in the build() method demonstrated earlier.