RL-Glue Matlab Codec 1.0 Manual

Brian Tanner :: brian@tannerpages.com

Contents

1   Introduction

This document describes how to use the Matlab RL-Glue Codec, a software library that provides socket-compatibility with the RL-Glue Reinforcement Learning software library. Matlab is a brand new codec, created specifically for the RL-Glue 3.0 release. Special thanks (or pokes) should go to Dale and Doina who gave voices to the countless others who probably have scorned us for not supporting Matlab earlier. It turned out to not even be that difficult :)

For general information and motivation about the RL-Glue[1] project, please refer to the documentation provided with that project.

This codec will allow you to create agents, environments, and experiment programs in Matlab.

This software project is licensed under the Apache-2.0[2] license. We're not lawyers, but our intention is that this code should be used however it is useful. We'd appreciate to hear what you're using it for, and to get credit if appropriate.

This project has a home here:
`http://rl-glue-ext.googlecode.com`

---

[1]`http://glue.rl-community.org/`
[2]`http://www.apache.org/licenses/LICENSE-2.0.html`

1.1 Software Requirements

To run agents, environments, and experiments created with this codec, you will need to have RL-Glue installed on your computer.

Compiling and running components with this codec requires Matlab. You will also need to have the RL-Glue Java Codec on your computer, this Matlab codec uses the Java codec's network capabilities.

Possible Contribution: Someone with Matlab experience could help us find out what version of Matlab is required to use this codec, and could help us update the codec to be as robust as possible to older versions.

1.2 Getting the Codec

The codec can be downloaded either as a tarball or can be checked out of the subversion repository where it is hosted.

The tarball distribution can be found here:
http://code.google.com/p/rl-glue-ext/downloads/list

To check the code out of subversion:
svn checkout http://rl-glue-ext.googlecode.com/svn/trunk/projects/codecs/Matlab Matlab-Codec

1.3 Installing the Codec

There is no real "installation" for the codec per-se. In each Matlab session, you'll have to add the codec source files to your Matlab path

To add the Matlab codec source files to your path, and to add the Java codec to your Matlab Java path, I do the following: Matlab will want to know where the codec source files are, so we'll frequently use code like:

```
>> clear java;
>> javaaddpath('/path/to/my/java/codec/products/JavaRLGlueCodec.jar')
>> addpath(sprintf('%s%s',pwd(),'/to/matlab/codec/src/glue'));
>> addpath(sprintf('%s%s',pwd(),'/to/matlab/codec/src/agent'));
>> addpath(sprintf('%s%s',pwd(),'/to/matlab/codec/src/environment'));
```

You can make your life easier by writing a script that does this. There is probably also a way to put these things in your default paths.

POSSIBLE CONTRIBUTION: IF someone wants to investigate the options for having a script "install" the Matlab modules into a system classpath, or anything else to make this more convenient,

we would like to have your help!

2 Gotchas!

2.1 Disconnect your Agent/Environment/Experiment

If your agent/environment/experiment crashes or terminates prematurely, and the `rl_glue` executable socket server is still running on your machine, you should manually disconnect your Matlab component so that the `Java` socket gets cleaned up. Depending on what you were running, the code would be:

```
>> disconnectAgent()            %For agents
>> disconnectEnvironment()      %For environments
>> disconnectGlue()             %For experiments
```

If you don't disconnect manually in these cases, Matlab can get hung up and you'll have to restart it.

2.2 One Component At A Time

Matlab is inherently single threaded, so without making the codec overly complicated, we had to limit each Matlab instance to running a single component at at time. This means that you can only run ONE of an agent OR environment OR experiment in Matlab at a time. If you know how to open multiple Matlab instances (not sure if this is possible), you can run one component per instance. We think that the usual case will probably be to run your agent in Matlab, but run the experiment and environment in another language, like Java, C, or Python.

We may lift this restriction in the future, or write a local glue engine so that all three can run together inside Matlab.

3 Agents

We have provided a skeleton agent with the codec that is a good starting point for agents that you may write in the future. It implements all the required functions and provides a good example of how create and run a simple agent.

The pertinent file is:

```
examples/skeleton_agent/skeleton_agent_construct.m
examples/skeleton_agent/skeleton_agent_init.m
```

```
examples/skeleton_agent/skeleton_agent_start.m
examples/skeleton_agent/skeleton_agent_step.m
examples/skeleton_agent/skeleton_agent_message.m
examples/skeleton_agent/skeleton_agent_end.m
examples/skeleton_agent/skeleton_agent_cleanup.m
```

You may realize that this feels like many files to create a simple agent. You're right. Usually, in Matlab, it's important to have each of your public functions in a separate file so that you can easily call the functions from the global Matlab scope. In Section 4 we'll show how you can create agents and environments using a single file because of our function pointer strategy.

This agent does not learn anything and randomly chooses integer action 0 or 1.

You can compile and run the agent like:

```
>$ cd examples/skeleton_agent/
>$ theAgent=skeleton_agent_construct();
>$ runAgent(theAgent);
```

skeleton_agent_construct() creates a struct with function pointers to the other skeleton_agent methods. runAgent(theAgent) then connects to RL-Glue and runs one step at a time until RL-Glue disconnects.

Alternatively, for a more interactive experience, you can run the agent manually one step at a time:

```
>$ cd examples/skeleton_agent/
>$ theAgent=skeleton_agent_construct()
>$ connectAgent(theAgent);
>$ runAgentLoop(theAgent);     %run one step
>$ runAgentLoop(theAgent);     %run one step
>$ runAgentLoop(theAgent);     %run one step
   ...
```

Using this method, you can stop and examine what your agent is learning, and potentially modify, visualize, or analyze it however you like.

You will see something like:

```
>> runAgent(theAgent)
RL-Glue Matlab Agent Codec Version: 1.0 ($Revision: 314 $)
    Connecting to rl_glue at host: localhost on port 4096
```

This means that the skeleton_agent is running, and trying to connect to the rl_glue executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-c` on your keyboard, and then starting `rl_glue` on your local machine (this will cause Matlab to come back from Java, and process your kill command). Don't forget to `disconnectAgent()`.

We're sorry that killing an agent (or environment) is so kludgy. This is the first version of the Matlab codec, so there are some kinks to get out. We welcome your expertise in helping make this codec better!

The Skeleton agent is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

POSSIBLE CONTRIBUTION: If you take a look at the agent and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

4   Environments

We have provided a skeleton environment with the codec that is a good starting point for environments that you may write in the future. It implements all the required functions and provides a good example of how to compile a simple environment. This section will follow the same pattern as the agent version (Section 3). This section will be less detailed because many ideas are similar or identical.

The pertinent file is:


```
examples/skeleton_environment/skeleton_environment.m
```


This environment is episodic, with 21 states, labeled $\{0, 1, \ldots, 19, 20\}$. States $\{0, 20\}$ are terminal and return rewards of $\{-1, +1\}$ respectively. The other states return reward of 0. There are two actions, $\{0, 1\}$. Action 0 decrements the state number, and action 1 increments it. The environment starts in state 10.

You can compile and run the environment like:


```
>$ cd examples/skeleton_environment
>$ theEnv=skeleton_environment();
>$ runEnvironment(theEnv);
```


You will see something like:


```
RL-Glue Matlab Environment Codec Version: 1.0 ($Revision: 314 $)
    Connecting to rl_glue at host: localhost on port 4096
```

This means that the skeleton_environment is running, and trying to connect to the `rl_glue` executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-c` on your keyboard, and then starting `rl_glue` on your local machine (this will cause Matlab to come back from Java, and process your kill command). Don't forget to `disconnectEnvironment()`.

The Skeleton environment is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

POSSIBLE CONTRIBUTION: If you take a look at the environment and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

5 Experiments

We have provided a skeleton experiment with the codec that is a good starting point for experiment that you may write in the future. It implements all the required functions and provides a good example of how to compile a simple experiment. This section will follow the same pattern as the agent version (Section 3). This section will be less detailed because many ideas are similar or identical.

The pertinent files are:

```
examples/skeleton_experiment/skeleton_experiment.m
```

This experiment runs `RL_Episode` a few times, sends some messages to the agent and environment, and then steps through one episode using `RL_step`.

```
>$ cd examples/skeleton
>$ skeleton_experiment();
```

You will see something like:

```
Experiment starting up!
RL-Glue Matlab Experiment Codec Version: 1.0 ($Revision: 314 $)
    Connecting to rl_glue at host: 127.0.0.1 on port 4096
```

This means that the skeleton_experiment is running, and trying to connect to the `rl_glue` executable server on the local machine through port 4096!

You can kill the process by pressing `CTRL-c` on your keyboard, and then starting `rl_glue` on your local machine (this will cause Matlab to come back from Java, and process your kill command). Don't forget to `disconnectGlue()`.

The Skeleton experiment is very simple and well documented, so we won't spend any more time talking about it in these instructions. Please open it up and take a look.

POSSIBLE CONTRIBUTION: If you take a look at the experiment and you think it's not easy to understand, think it could be better documented, or just that it should do some fancier things, let us know and we'll be happy to do it!

6   Putting it all together

At this point, we've run each of the three components, now it's time to run them with the `rl_glue` executable server. As mentioned elsewhere in this document, in Matlab we can only run ONE of the agent, experiment, or environment at a time. So, for this example we'll run the `Java SkeletonAgent` and `Java SkeletonEnvironment` with the `Matlab skeleton_experiment`. You could really run any agent and environment of your choice.

The following will work if you have RL-Glue installed in the default location, and the Java codec downloaded and saved to the location described in the Java codec manual.

For this, we'll use a separate terminal window in addition to the Matlab interpreter. In a separate terminal window, do:

```
>$ cd /path/to/Java/codec/examples/skeleton
>$ rl_glue &
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonAgent &
>$ java -classpath ~/JavaCodec/JavaRLGlueCodec.jar:. SkeletonEnvironment &
```

In the Matlab interpreter:

```
>$ cd examples/skeleton_experiment
>$ skeleton_experiment();
```

If RL-Glue is not installed in the default location, you'll have to start the `rl_glue` executable server using it's full path (unless it's in your `PATH` environment variable):

```
>$ /path/to/rl-glue/bin/rl_glue &
```

In the Matlab window, you should see the following if it worked:

```
>> skeleton_experiment()
Experiment starting up!
RL-Glue Matlab Experiment Codec Version: 1.0 ($Revision: 314 $)
    Connecting to rl_glue at host: 127.0.0.1 on port 4096
```

8

```
     Experiment Codec Connected
RL_init called, the environment sent task spec: 2:e:1_[i]_[0,20]:1_[i]_[0,1]:[-1,1]


----------Sending some sample messages----------
Agent responded to 'what is your name?' with: my name is skeleton_agent, Java edition!
Agent responded to 'If at first you do't succeed; call it version 1.0  '
with: I don't know how to respond to your message

Environment responded to 'what is your name?' with: my name is skeleton_environment, Java edit:
Environment responded to 'If at first you don't succeed; call it version 1.0  '
with: I don't know how to respond to your message


----------Running a few episodes----------
Episode 0  10 steps   -1.000000 total reward  natural end 1
Episode 1  10 steps   -1.000000 total reward  natural end 1
Episode 2  10 steps   -1.000000 total reward  natural end 1
Episode 3  10 steps   -1.000000 total reward  natural end 1
Episode 4  10 steps   -1.000000 total reward  natural end 1
Episode 5  1 steps    0.000000 total reward  natural end 0
Episode 6  10 steps   -1.000000 total reward  natural end 1


----------Stepping through an episode----------
First observation and action were: 10 and: 0


----------Summary----------
It ran for 10, total reward was: -1.000000
```

7   Who creates and frees memory?

The RL-Glue technical manual has a section called *Who creates and frees memory?*. The general approach recommended there is to make a copy of data you want to keep beyond the method it was given to you. The same rules of thumb from that manual should be followed when using the Matlab codec.

8   Advanced Features

This section will explain how to set custom target IP addresses (to connect over the network) and custom ports (to run multiple experiments on one machine or to avoid firewall issues).

Someone should write this later (Opportunity to contribute!).

9 Codec Specification Reference

This section will explain how the RL-Glue types and functions are defined for this codec. This isn't meant to be the most exciting section of this document, but it will be handy.

Instead of re-creating information that is readily available in the MatlabDocs, we will give pointers were appropriate.

9.1 Types

9.1.1 Simple Types

Unlike the C/C++ codec, we will not be using `typedef` statements to create special labels for the types. Since Matlab is loosely typed, these things aren't so hard and fast:

- *reward* is `double`

- *terminal* is `int` (1 for terminal, 0 for non-terminal) We hope to replace these with boolean eventually.

- *messages* come as Java `strings` and can be returned as Matlab strings

- *task specifications* come as Java `strings` and can be returned as Matlab strings

9.1.2 Structure Types

All of the major structure types (observations, actions) come off the network as the appropriate object from the Java codec. The Java codec manual should have all the information required to understand those objects.

So in a given Matlab method, like
`function theAction=skeleton_agent_step(theReward, theObservation)`, theObservation is actually of type: `org.rlcommunity.rlglue.codec.types.Observation`.

Java and Matlab play very well together, so you can do things like:

```
>> testObs=org.rlcommunity.rlglue.codec.types.Observation();
>> testObs.intArray=[1 2 3 4];
>> testObs.doubleArray=[0.1 0.5];
>> testObs.charArray='fun things!';
```

```
>> testObs.toString()

ans =

numInts: 4
numDoubles: 2
numChars: 11
 1 2 3 4 0.1 0.5 f u n   t h i n g s !
```

9.2   Functions

9.2.1   Agent Functions

All agent constructor functions **should set** the same functions as our Skeleton agent.

Useful utility methods for connecting, disconnecting, and running with the `rl_glue` executable server are in the `agent` directory of the Matlab codec source.

9.2.2   Environment Functions

All environment constructor functions **should set** the same functions as our Skeleton environment.

Useful utility methods for connecting, disconnecting, and running with the `rl_glue` executable server are in the `environment` directory of the Matlab codec source.

9.2.3   Experiments Functions

All experiments **can call** the methods in the `glue` directory. In this case we'll include their prototypes, because the source file is full of implementation details.

10   Frequently Asked Questions

We're waiting to hear your questions!

10.1   Where can I get more help?

10.1.1   Online FAQ

We suggest checking out the online RL-Glue C/C++ Codec FAQ:
`http://glue.rl-community.org/Home/Extensions/matlab-codec#TOC-Frequently-Asked-Questions`

The online FAQ may be more current than this document, which may have been distributed some time ago.

10.1.2   Google Group / Mailing List

First, you should join the RL-Glue Google Group Mailing List:
`http://groups.google.com/group/rl-glue`

We're happy to answer any questions about RL-Glue. Of course, try to search through previous messages first in case your question has been answered before.

10.2   What's up with the Task Spec Parser?

We've distributed the existing task spec parser from the previous version of RL-Glue with this codec. However, it is not officially supported as we are trying to ratify a new and improved task specification language.

We will release an updated codec with certified task spec parser once the new language has been ratified.

For more information, please go here:
`http://glue.rl-community.org/Home/rl-glue/task-spec-language`

11   Credits and Acknowledgements

Brian Tanner wrote the Matlab codec. Yay me.

11.1   Contributing

If you would like to become a member of this project and contribute updates/changes to the code, please send a message to rl-glue@googlegroups.com.

Document Information

```
Revision Number: $Rev: 314 $
Last Updated By: $Author: brian@tannerpages.com $
Last Updated   : $Date: 2008-10-05 22:31:32 -0600 (Sun, 05 Oct 2008) $
$URL: https://rl-glue-ext.googlecode.com/svn/trunk/projects/codecs/Matlab/docs/MatlabCodec.tex
```