CS7637: Project Submission Instructions (Fall 2016)

This fall, we're bringing back the server-side autograder for CS7637 projects. This autograder brings with it a number of advantages:

- You'll receive instant feedback on whether or not your code ran in our environment.
- You'll be able to run your agent against the unseen problems in advance, getting a limited amount of general feedback on its performance so far instead of waiting for feedback from the graders. (This should especially help if you want to build a highly experimental agent but are a little worried about how it will perform in practice.)
- Rather than running all the agents en masse after the deadline, we'll simply download the results after the deadline.
- You'll be able to take advantage of cloud computing resources instead of committing your computer to running long algorithms.

Traditionally, evaluating projects has been one grader's entire job (20 hours/week) all semester. This semester, we're hoping this new approach cuts down that time significantly while also getting you grades and feedback more quickly.

All that said, this is still an experimental system. Things could go horribly wrong. If so, don't panic: post to the `autograder` folder on Piazza to alert us to any problems. We will absolutely take into consideration bumps along the way and ensure they do not affect your grade.

Submission For Grading

As far as getting a grade is concerned, what you need to know is: you should make sure to submit your code to the autograder before the deadline. We recommend you do so more than once to get feedback and improve your agent, but once is the minimum to get graded. The autograder replaces the T-Square submission flow for the agent code, so the T-Square submission will **only** require your report PDF.

The portion of the project grade based on agent performance will be calculated from your agent's best overall run. So, for instance, if you run your agent twice, scoring 8/12 on basic and 6/12 on test then dropping to 8/12 on basic and 4/12 on test, the 8/12 and 6/12 score will hold.

Getting Started

When you're getting started, you should still primarily consult the Overall Project Guidelines and the Project 1 description. Download the code, work on it on your local computer, and run it against the problems shared with the project framework. The instructions provided here are for submitting your code to the autograder, which you should only do once you're reasonably comfortable with how it's performing locally.

The autograder gives you feedback that you can use to improve your agent going forward. So, don't just wait until the last day to submit: try out your agent earlier and see how it's doing. (An early submission, like an early T-Square submission, also guards you against last-minute computer failures and network outages.) However, because the number of submissions is capped the autograder isn't like a Udacity code quiz: don't solely rely on the autograder to debug and run your code.

Finally, we strongly recommend taking advantage of the error-check "assignment" to check for and debug agent exceptions and submission errors. The error-check will run your agent against all problem sets for the current project iteration. While it will only give detailed feedback on problem sets you can yourself run locally, the error-check is very useful for making sure you have included all necessary agent files, configured your language.txt file correctly (if applicable), and so forth.

Submission Instructions

Below are the step-by-step submission instructions for submitting your agent to the autograder.

1. Make sure your agent is built using the official framework! You can obtain it in either of two ways.
    - To clone the git repository, use:
      `'git clone --recursive https://github.gatech.edu/Dilab/KBAI-package-java.git '`
      for Java, or
      `'git clone --recursive https://github.gatech.edu/Dilab/KBAI-package-python.git '`
      for Python. This makes it easier to 'git pull' any (rare) framework changes or fixes that must be made after the project is released.
    - Or, you may download Project-Code-Java or Project-Code-Python as .zip files from this folder. This method allows you to obtain the code if you are having trouble accessing the Georgia Tech Github site.
2. Install a couple of packages required for running the submit script.
    - `pip install requests future`
3. In your project directory (for either language) you will find a Python script called submit.py. Run it, making sure to specify "--provider gt" and the project number you're trying to submit. We also strongly recommend you first run the error check:

```
python submit.py --provider gt --assignment error-check
```

- If you are using Python 3, please update the language.txt file to say 'python3' instead of 'python'.
- If you have created additional files, make sure to include them with the --files parameter. For example, if you are submitting project 1 built in Java and your agent uses two extra classes called RavensTransform and RavensNetwork, you would run:

  ```
  python submit.py --provider gt --assignment P1 --files
       RavensTransform.java RavensNetwork.java
  ```

  Note that it is not necessary to explicitly list all the files included in the project framework.

4. The script will prompt you for your login information. Make sure to use your Georgia Tech login credentials so that we can credit you for your submission! Optionally, you can save your login information when the script prompts you to "save jwt" -- this will avoid the need to re-enter it on future runs.
   - Note: if you are using two-factor authentication, an additional step is required to allow the submit script to log in on your behalf -- please see the instructions at https://bonnie.udacity.com/auth_tokens/two_factor.
5. The script will run for a while and then return a set of results which looks something like this:
   - Problem,Correct?,Correct Answer,Agent's Answer
   - "Challenge Problem B-04",0,4,1
   - "Basic Problem B-12",1,1,1
   - ...

   It will also display some summary information per problem set in the console.
6. You may find it convenient to open the CSV-formatted results in a spreadsheet program like Excel for easy browsing and reference.

If you encounter any problems at all when running the autograder, please post to Piazza in the `autograder` folder. This helps make sure we'll see your issue and respond promptly!

Submission Throttling

Be aware: you are limited in the number of times you can submit to the autograder! For Project 1, you may submit **5 times total**. That means that you likely want to get an early start on the project to take full advantage of the ability to refine your agent.

This throttle exists for a number of reasons. Pragmatically, it is to ensure the servers aren't overly strained by running the same agents over and over again. It's also to ensure agents do not overfit based on feedback received from the Test set; with lots of attempts, one could infer lots of information from the set even without seeing any output with lots of runs.

However, more importantly this throttling replicates something very important about the human intelligence test experience. When studying for tests, humans can take lots of time to practice on sample problems, but the number of real attempts are relatively limited. Humans don't simply get to take an intelligence test over and over until they get a great score, but at the same time, humans do have the opportunity to take real tests multiple times and learn from the experience each time. This arrangement aims to replicate that: your agent can practice an unlimited number of times on the Basic and Challenge problems and any sample problems you devise, but it has a scarce number of attempts at the real problems.

Output

The autograder will not display any output from your code except for the final results. This is to prevent agents from disclosing the contents of the hidden problems. Agents are permitted to create their own external files if need be, but the contents of those files will not be available after submission either. Similarly, agents will not be able to write to any other files outside the autograder server.

In other words: just like a real standardized test, your agent is not permitted to remove any testing materials from the testing "facility".

Submission Errors

If your submission errors out, you'll see a result indicating what kind of error the code encountered. (For Java, this will be either a build or an execution error; since Python is an interpreted language, only execution errors are possible.)

Here are some debugging steps you can try:

- If you have a build error in Java, try running 'cd workspace/ravensproject; javac @sources.txt' to build at the command line. This is the same command we use to build your code, and running it should give you an exact file/line where the build error occurs.
- If you have an execution error in Python, make sure that your language.txt file reflects the actual Python version you are running locally. If you're using an IDE like IntelliJ, your project settings should tell you what interpreter you're using. If you're running at the command line, you can find the interpreter version with 'python --version'. Usually, if you're running 'python' it's Python 2, and if you're running 'python3' it's Python 3. Similarly, you should update language.txt to say 'python' for Python 2 and 'python3' for Python 3.

- If your agent results are different than expected (e.g., you get a different score on Basic and Challenge sets when you run locally), double-check the language.txt again! If that's correct, check to see if your local library versions match the ones on the autograder; those are:
  - Java: JDK 8u73
  - Python 3: Python 3.4.3
  - Python 2: Python 2.7.11
  - Pillow: Pillow 3.1.1
  - Numpy: 1.10
- If you get a message containing the text "The total output length exceeded the limit of 65536 bytes.", your agent is dumping too much output to stdout. Try removing print/log statements before you resubmit.
- If you include a class named RavensSolver and get an error about it, rename it to something else and resubmit - this class name is used by the autograder.
- If the submission script hangs without asking you for credentials, or after asking you for your username but before asking you for your password, check what shell you're using. We have seen this issue occasionally with Windows users who are attempting to submit from the git shell or Cygwin. If you are using one of these, try resubmitting from the standard command prompt.

If all else fails, please post on Piazza and include the text of any error message you find with the above steps - your classmates or TAs can help you figure out where things might be going awry!

Effective Deadlines

Your agents must be *submitted* before the deadline, 11:59PM UTC-12. However, it's alright if your agent is still running after the deadline: you simply need to worry about having the agent submitted before then. There's a little lag built into the system so if you encounter some difficulties right near the deadline, you should be fine.

Remember, you're only permitted 5 executions total. Your 5th submission will be your final attempt, and you will not be able to revise and resubmit. We strongly, ***STRONGLY*** encourage submitting to the error-check script before each of these 5 submissions to make sure that there are no unpleasant surprises. You can check the number of submissions you have remaining (as well as some other fun statistics about your agent) at https://bonnie.udacity.com/.