Milestone 2 Progress Evaluation

Automatic Program Feedback

1.

Team Members:

Andre Leone     Ryan Hartman     Calvin Winget
aleone2013@my.fit.edu  rhartman2014@my.fit.edu  cwinget2014@my.fit.edu

Faculty sponsors:

Dr. Philip Chan     Dr. Ryan Stansifer
pke@cs.fit.edu     ryan@cs.fit.edu
2. Progress of current milestone (progress matrix)

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion %</th>
<th>Andre Leone</th>
<th>Ryan Hartman</th>
<th>Calvin Winget</th>
<th>To do</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Build version 1 of the feedback program</td>
<td>100%</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
<td>none</td>
</tr>
<tr>
<td>2. Evaluate Results</td>
<td>100%</td>
<td>50%</td>
<td>25%</td>
<td>25%</td>
<td>none</td>
</tr>
<tr>
<td>3. Analyze incorrect feedback</td>
<td>100%</td>
<td>30%</td>
<td>40%</td>
<td>30%</td>
<td>Analyze more after we get more data to run the feedback program with</td>
</tr>
<tr>
<td>4. Make updates to the automated feedback program</td>
<td>60%</td>
<td>20%</td>
<td>20%</td>
<td>20%</td>
<td>Continue this in the next milestone if we come across more things that we could modify</td>
</tr>
</tbody>
</table>

3. Discussion of Tasks

- Task 1: We have finished the initial framework that will be used for all iterations of our current programs. Our current design separates the program into three parts. The first part is responsible for recording program behavior. This part takes the reference and submitted programs and compiles them. With them compiled they are then instrumented so that their behaviors can be recorded. Finally, it then runs the programs and saves the generated output into trace files. The paths for these files are then given to the second portion of the program. This part is responsible for generating the feedback. Our current implementation for this takes the trace files generated and compares each submitted program to all of the reference programs and computes the edit distance between their
trace files. This process is done for each test case. The submitted program is then assigned the same feedback as whichever reference program is has the lowest total edit distance with. With the feedback now generated for each program, the evaluation step can be done. For the automated evaluation, the program runs through each assigned feedback and verifies if the assigned feedback matches the expected feedback. The percentage of correct feedback assigned is then calculated and outputted.

- Task 2: On our first experiment, we used 5 reference programs and 6 submitted programs. 2 out of 6 submitted programs were assigned correct feedback (33%).
- Task 3: For our initial tests on the selection sort programs provided by Dr. Stansifer as well as some written ourselves, We correctly identified 2 out of the 6 submission programs. These two programs were both correctly identified as being correct. For the feedback that was generated incorrectly, one reason for this is that some programs do not have a reference program that has the same bug as them. This then means that it would be impossible to generate correct feedback. We have received more selection sort programs from Dr. Stansifer and we are going to label these and also make sure each submitted program has a reference program it can match with, and also that each reference program has at least one submitted program with the same bug.
- Task 4: One of our initial changes to version 1 of the feedback system was to make the system load variables from a main.config files. Using the config file allows us to run different experiments without modifying any code. Another change that we made after we built version 1 was to add variables “recompile”, “re-instrument”, and “retrace”. These variables allow us to speed up experiments after we have already done one experiment on the dataset. As long as a future experiment does not modify how the class files are instrumented, we do not need to compile, instrument, or generate the trace files again.

4. Discussion of contributions

- Andre Leone:
  - I worked on the record behaviors portion of version 1 of the feedback system. The record behaviors consisted of three inner parts. The first part of record behaviors was to compile all of the .java programs. This includes both the reference and submitted java programs. The second part of the record behaviors was to instrument the class files for both the reference and submitted java programs. The instrumentation was doing by making external shell calls to Dr. Stansifer’s instrumentation program. Lastly, all of the instrumented class files were run for each test case that we have for that problem. Trace files were outputted and saved to each of the corresponding directories.
  - In addition to participating in the initial build of version 1 of the system, I also helped with analyzing why the feedback was incorrect and making updates to the system to make it easier for future experiments.

- Ryan Hartman:
  - I wrote the feedback generation portion of the code for this milestone. It calculates the edit distance between the reference and submitted program for
each test case and maintains the distance between them for each. With these distances, the program then classifies each of the submitted program using a nearest neighbor classifier approach.

- I also investigated if the instrumentation we were doing worked on all calls for set and get methods on all classes which implement the list interface, or if it is just restricted to arraylists. I found that it works for all list classes.

- Calvin Winget:
  - I worked on the evaluate feedback portion of version 1 of the feedback program. The evaluate feedback portion consisted of interpreting the output from the generate feedback and checking whether or not the generated feedback matched the expected feedback from the summary.txt file. Also, another evaluation method was implemented to compare Hackerrank score files instead of feedback files to see whether or not Hackerrank problems were paired with the correct scores. Lastly, the evaluation portion was setup to output (via JSON) the feedback and the expected feedback pairings, along with the two folder paths of each submitted program that was paired with a reference program. This provides for a simple way to visually analyze how well the generate feedback portion is doing.
  - Investigated why some trace files were showing up empty.
  - Checked feedback to see if it actually matched the bugs in the program.
  - Started work on making sure we have 2 submissions for every reference program.

5. Plan for the next Milestone (task matrix)

<table>
<thead>
<tr>
<th>Task</th>
<th>Andre Leone</th>
<th>Ryan Hartman</th>
<th>Calvin Winget</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Build version 2 of the feedback program</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>2. Evaluate Results</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>3. Analyze incorrect feedback</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
<tr>
<td>4. Brainstorm possible updates that can be done to improve the feedback results</td>
<td>33%</td>
<td>33%</td>
<td>33%</td>
</tr>
</tbody>
</table>
6. Discussion of each planned task for the next milestone

- Task 1: For version two, we are going to continue to use the same base structure that we created for version one. Some of the changes we are looking into doing are changing the weighting for edit distance. This would mean making some operations more costly than others. Another approach we are going to try is instead of just using the total sum of the edit distance, we are going to investigate other possible ways to incorporate all test cases to see if they provide better results.
- Task 2: The evaluation we have in place now should just be able to be re-run to determine the accuracy of our classification.
- Task 3: To analyze the incorrect feedback we will look at which feedback was incorrect and then look at what the feedback should have been. We will look at what other programs the incorrect feedback program is clustered near and brainstorm possible reasons why.
- Task 4: After knowing what feedback was provided incorrect and why, we will then brainstorm changes to our algorithm to prevent these types of mistakes from happening again in future experiments. The improvements to the algorithm will be tested in future experiments.

7. Sponsor feedback on each task for the current milestone

- Task 1:

- Task 2:

- Task 3:

- Task 4:

Sponsor Signature: _______________________________ Date: ___________________
8. Sponsor Evaluation

- Sponsor: Detach and return this page to Dr. Shoaff
- Score (0-10) for each member: circle a score (or circle two adjacent scores for .25 or right down a real number between 0 and 10)

<table>
<thead>
<tr>
<th>Member</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>5.5</th>
<th>6</th>
<th>6.5</th>
<th>7</th>
<th>7.5</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andre Leone</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
<tr>
<td>Calvin Winget</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
<tr>
<td>Ryan Hartman</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>5.5</td>
<td>6</td>
<td>6.5</td>
<td>7</td>
<td>7.5</td>
<td>8</td>
<td>8.5</td>
<td>9</td>
<td>9.5</td>
<td>10</td>
</tr>
</tbody>
</table>

- Sponsor Signature: ___________________________ Date: ______________