Misconduct Detection Project(MDP) Tool Testing and Debugging Report No.1

Yucheng Xie
August 22, 2018
Supervisor: Dr. Kyriakos Kalorkoti
Master of Science
School of Informatics
University of Edinburgh
Software Version: 1.0

Contents

1 Summary

This "testing and debugging report" is used as a record for later developers referring. Each bug in this report has been illustrated by following three aspects:

1. The details of the bug. This part is mainly giving the details of the bug, including how to reproduce the bug.
2. The analysis of the bug. This part is mainly focusing on analyzing the bug and providing fixing solutions to the bug.
3. The test cases of the bug. This part is providing the test cases for verifying the bug is resolved.

Especially, in the above information, later developers should always rerun the test cases of a bug if any code related to the bug is changed. Rerunning test cases could guarantee that later changes on the code are not making these bugs recur.

Involved bugs in this report:

1. DET-001: Probability Calculation Error
2. DET-002: JPlag Results Interpreting Error

2 DET-001: Probability Calculation Error

2.1 Bug Report Part

General Information

Title	Probability Calculation Error
ID	(detection package) DET-001
Priority	Critical

Description

When duplicating two identical files in the source folder, some ridiculous probability results such as 2 can be obtained.

How to Reproduce

1. Set up the source folder with only 1 submission.
2. Duplicate the submission in the source folder. Now, we should have two identical submissions in the source folder.
3. Select one segment in the source file. Here we select the first function in the source file. However, selection of segments should not affect reproducing this bug.
4. Perform detection by clicking "Next" button on the "Selection" page.

Screenshots

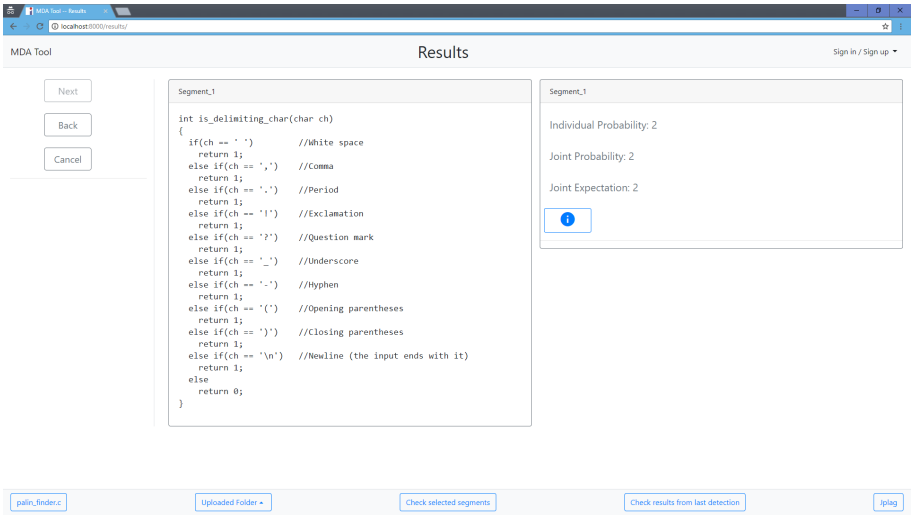

Figure 1: DET-001: Probability Calculation Error screenshot

2.2 Bug Analysis Part

As a result, we made a calculation error here in the code. We did not exclude the source file itself from the folder, which causes counts to be "2" in this case. We then exclude the source file from the source folder.

2.3 Test Cases Part

Test Case 1:

1. Set up a source folder, which contains exactly one submission.
2. Duplicate this submission, make sure the source folder contains two identical submission.
3. Select several segments randomly and perform detection.

Expected Results for this Test Case:

The results probabilities (both individual and joint) of all segments should always be "1".

3 DET-002: JPlag Results Interpreting Error

3.1 Bug Report Part

General Information

Title	JPlag Results Interpreting Error
ID	DET-002
Priority	Critical

Description

The result probabilities in some cases might become "0". This bug is cased by an error in interpreting results obtained from JPlag.

How to Reproduce

1. Set up the source folder with only 1 submission.
2. Duplicate the submission in the source folder. Now, we should have two identical submissions in the source folder.
3. Select *More Than* one segments in the source file. Here we select the first two functions in the source file. However, which functions are selected should not affect reproducing this bug.
4. Perform detection by clicking “Next” button on the “Selection” page.

Screenshots

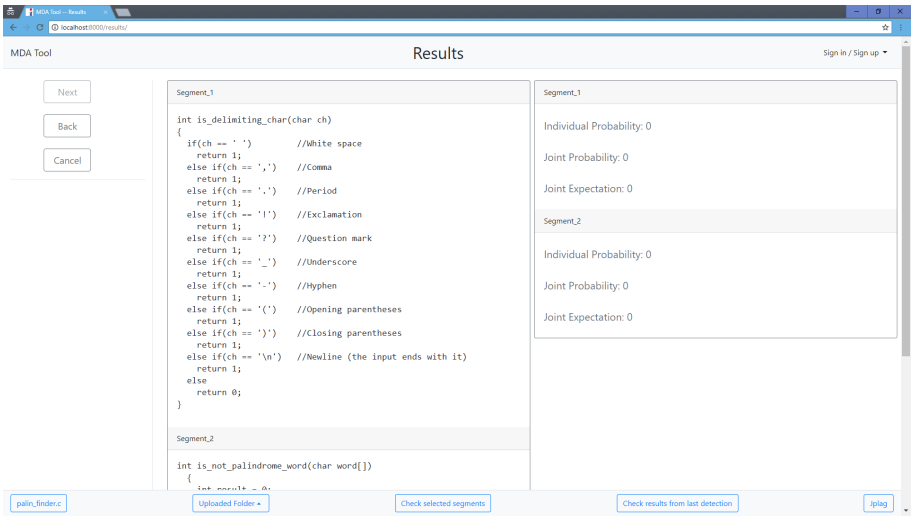

Figure 2: DET-002: JPlag Results Interpreting Error screenshot

3.2 Bug Analysis Part

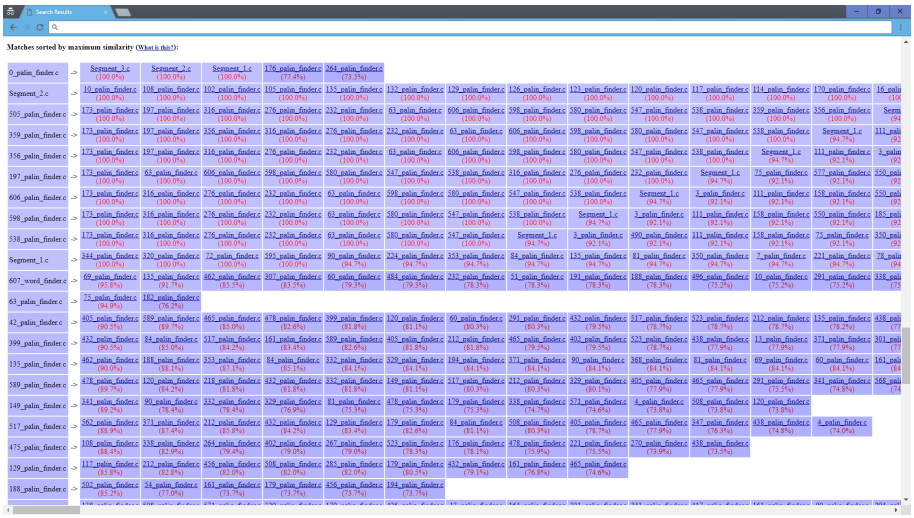

Figure 3: JPlag result index page screenshot

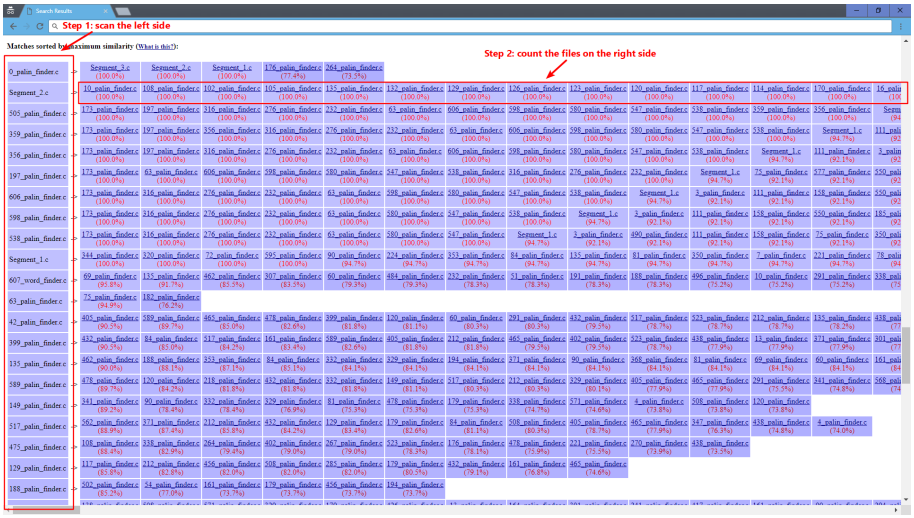

Figure 4: JPlag result index page screenshot with marks

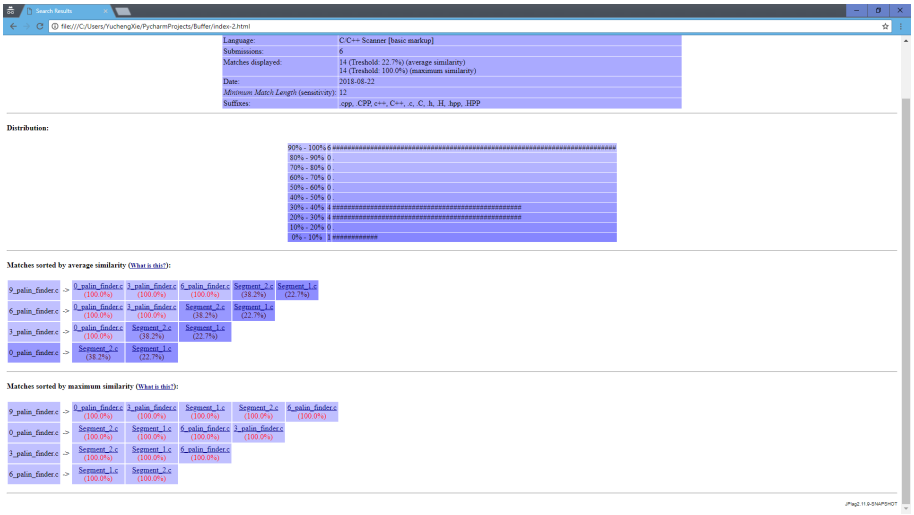

Figure 5: JPlag result index page which cases error screenshot

To illustrate how this bug is occurred, we need to mention how we extract information based on JPlag's results. We followed the following steps to obtain information from JPlag's results:

For example, the JPlag result index page usually looks like as shown in Figure 3. On this page, we scan left side first. Then, if we find any "Segment" on the left side, we just count the similar files on the right side. After that, we use this count number to calculate probabilities. (as shown in Figure 4)

However, this is assuming that the "segments" will appear on the left side. If the "segments" are less suspect than other files, it actually will not appear on the left side. Like the situation we illustrated above. Under this situation, the JPlag final result index page will look like Figure 5. In Figure 5, we notice that all "segments" are not appearing on the left side. This is the reason why the "0" probabilities occur.

To resolve this bug, we need to consider the "segments" even if they only

appear on the right side. We noticed that the JPlag will show every pairwise record for exactly one time. For example, assume that we have "file 1" and "file 2" are similar. If both of them are suspect enough to be shown on the left side, the record will appear exactly one time overall. That is, if this record appears at the right side of "file 1", it then will not appear at the right side of "file 2" and vice versa.

Therefore, here to fix this bug, our solution should be counting both situation for one file. That is, if our segment appears on the left side, append what appear on the right side to the overall record of this segment. Then, if this segment appears on the right side for other files, append this record to the overall record of this segment.

3.3 Test Cases Part

Test Case 1:

1. Set up the source folder with exactly 1 submission.
2. Duplicate the submission in the source folder for more than one time. Here we choose to duplicate for 3 times. Now, we should have four identical submissions in the source folder.
3. Select *More Than* one segments in the source file randomly.
4. Perform detection by clicking "Next" button on the "Selection" page.

Expected Results for this Test Case:

The results probabilities (both individual and joint) of all segments should always be "1". The joint expectation should be "3".

Appendix

A Explanation of Bug Priorities

In this report, we divided all bugs into three different priority levels. They are:

1. Critical:

   Something could case the whole system to be stopped or case the system producing wrong results.

2. Normal:

   Something might affect later developing. Or something might case some further problems if it is not properly resolved.

3. Minor:

   Annoying but not harmful to the functionalities. Such as layout displaying error in different browsers.