Anti-plagiarism readme

Eric Scott Freeman

2015-11-08

Contents

1 Supported Languages

Tool	Java	Go	C	C++	C#	Python	Perl	others
Moss	✓		✓	✓	✓	✓	✓	✓
JPlag	✓		✓	✓	✓			✓
dupl		✓						

Table 1: Officially supported languages by various tools

2 Getting the tools

2.1 Moss

Moss is a web service, while JPlag and dupl can be run locally. To access Moss, one must send an email to `moss@moss.stanford.edu` containing the following, with `<email address>` replaced with one's actual email address:

```
registeruser
mail <email address>
```

Moss will then send a Moss upload script with a unique user ID. Please note the user ID from the file.

```
$userid=1234567890;
```

The original script will not work with this application, because it does not expand the wildcard $*$. A new upload script is located in the base directory of this project. Please add your user ID to the new upload script.

2.2 dupl

To download and install dupl, run the following command:

```
go get -u github.com/mibk/dupl
```

dupl requires Go version 1.4 or higher.

2.3 JPlag

To download and install JPlag, get the code from `https://github.com/jplag/jplag`. Maven is also required. Download and install it if necessary. Before compiling the jar file, a small change needs to be made in order of JPlag to handle occurrences where a student's directory does not contain a lab. Inside the `jplag/jplag/jplag/src/main/java/jplag/Program.java` file, make the following changes at line 441. This is near the end of the `createSubmissions()` function. Comment out

```
throw new ExitException("Cannot find directory: " + file_dir.toString()
```

Add

```
continue;
```

After making the changes, go to the `jplag/jplag/jplag` directory inside the download and run the following command:

```
mvn clean generate-sources assembly:assembly
```

This should create a jar file inside the `./target/` directory called `jplag-x.y.z-SNAPSHOT-jar-with-dependencies.jar` where `x.y.z` is the specific version number of JPlag.

3 Tool commands

While it is not necessary to know the commands each anti-plagiarism tool uses, it may be helpful. The option described here are not the only options the tools use, but they are the options used by the application.

3.1 Moss

Here is an example of a Moss command:

```
./moss -l java -m 2 -d ./code/class01/student01/
    assignment01/*.java ./code/class01/student02/
    assignment01/*.java ./code/class01/student03/
    assignment01/*.java > assignment01.txt &
```

The first argument is the Moss upload script. The $-l$ flag signifies that the next argument will be the language the assignments were written in, which

in this case is Java. The $-m$ flag signifies that the following argument will be the threshold for Moss, which tells Moss to ignore matches that appear in more than this number of files. In this example, if a piece of code appears in more than 2 files, it is ignored. This is useful if instructors provide some functions or classes for their students to use. The $-d$ option signifies that directories will be compared instead of specific files. In this example, all the java files from three students' assignment 1 will be compared. Moss will search inside subdirectories. Finally the output from Moss is sent to a text file. The text file will contain a URL that has the results from Moss.

3.2 dupl

Here is an example of a dupl command:

```
dupl -t 15 -html ./code/class01/student01/assignment02/
    ./code/class01/student02/assignment02/ ./code/
   class01/student03/assignment02/ > assignment02.html
   &
```

The first argument is a call to dupl. The next argument, $-t$ is dupl's threshold. This is minimum nodes that pieces of code must be before dupl declares them as a duplicates. In this example, it is 15 nodes. $-html$ specifies html output. Next is a list of the directories. dupl will search inside subdirectories. Finally the output from dupl will be sent to an html file.

3.3 JPlag

Here is an example of a JPlag command:

```
java -jar ./jplag/jplag.jar -l java17 -t 15 -r ./
   results/lab1 -s lab1 ./students
```

The first three arguments say to run a Java jar file in that location called jplag.jar. $-l$ says which language to use, which in this case is Java 1.7. $-t$ is the minimum number of tokens to match (threshold) argument. The next argument, $-r$ specifies where to save the results. $-s$ says to check all the files in subdirectories with that label. So here any Java files in subdirectories labeled lab1 will be checked. The last argument is the base directory where the code resides.

4 Other software

4.1 goquery

goquery is needed for the application to parse html files. Please download it with the following command.

```
go get github.com/PuerkitoBio/goquery
```

Protobuf and gRPC are also required.

```
go get -u github.com/golang/protobuf/{proto,protoc-gen-go}

go get google.golang.org/grpc
```

5 Configuration file

There is a script located in the base directory of the anti-plagiarism project called *env.sh*. Please run the script.

```
source env.sh
```

LAB_FILES_BASE_DIRECTORY tells the application where to store the code pulled from GitHub. MOSS_USER_ID is the user ID found the Moss upload script. MOSS_FULLY_QUALIFIED_NAME contains the fully-qualified name of the Moss script. JPLAG_FULLY_QUALIFIED_NAME contains the fully-qualified name of the JPlag jar file. RESULTS_DIRECTORY tells the application where to store the results. MOSS_THRESHOLD is the Moss threshold described in the Tool commands subsection. DUPL_THRESHOLD and JPLAG_THRESHOLD contain similar values.

Here is an example:

```
export LAB_FILES_BASE_DIRECTORY=/home/autograde/repos
export MOSS_USER_ID=123456789
export MOSS_FULLY_QUALIFIED_NAME=/home/autograde/moss/
    moss
export JPLAG_FULLY_QUALIFIED_NAME=/home/autograde/go/
    src/github.com/jplag/jplag/jplag/target/jplag.jar
export RESULTS_DIRECTORY=/home/autograde/results
export MOSS_THRESHOLD=4
```

```
export DUPL_THRESHOLD=15
export JPLAG_THRESHOLD=15
```

6 Troubleshooting

If you get this error while trying to run the software:

```
# github.com/autograde/antiplagiarism/proto
proto/ap.pb.go:112: cannot use
    _Ap_CheckPlagiarism_Handler (type func(interface {},
     context.Context, func(interface {}) error) (
    interface {}, error)) as type grpc.methodHandler in
    field value
```

It probably means that the *ap.proto* file needs to be recompiled to work with a newer *gRPC* version. To recompile:

1. Make sure the following packages are installed: *autoconf*, *automake*, *libtool*, *gcc*, *g++*, *unzip*
2. Download *https://github.com/google/protobuf/archive/v*3.0.0−*alpha*−3.1.*tar.gz*
3. `tar -xvf protobuf-v3.0.0-alpha-3.1.tar.gz`
4. `cd protobuf-v3.0.0-alpha-3.1`
5. `./autogen.sh`
6. `./configure`
7. `make`
8. `sudo make install`
9. Go to the *github.com/autograde/antiplagiarism/proto* directory.
10. `protoc --go_out=plugins=grpc:. ap.proto`