Crowdsourcing Back-End (Weka) Specifications

I. Summary

After a user completes the survey, the computer will make an automated prediction on the final page. In order to make this prediction, we can plug the user's responses to the survey questions into a linear regression equation. To obtain a linear regression equation, all user data will need to be retrieved from the database and sent to Weka, which is a machine-learning algorithm written in Java. Weka will return the coefficients of the equation and constant (y-intercept). The coefficients can be stored in the database in tblQuestion as a new attribute called fldCoefficient. Each coefficient will correspond to a specific question in the database. The constant (y-intercept) will need to be stored in a separate table in the database. This table can be called tblConstant and have one attribute called fldConstant. There are many ways to go about designing the back-end of this project. For my project (predicting a user's personal savings), I created a php file that would retrieve the data from the database and write the data to an external file. Then the php file would execute a java file that called Weka and returned the linear regression equation coefficients. The php file would retrieve these coefficients and store them in the database. This plan accomplished these tasks perfectly, but failed to work with a cron job that would execute the php file periodically. I believe that the reason that the cron job will not work with this strategy is that the shell_exec php function that executes a java jar file is disabled in the UVM environment. However, the php and java files will still execute when the php file is run from the command line.

II. Development Process

I will now outline a step-by-step guide to creating the back-end of this project. This example entails a php file that retrieves data from a MySQL database, writes the data to an external file, and executes a java program that calls Weka. Weka analyzes the data and creates a linear regression equation. The java program will return the coefficients of this equation and the php file will store the coefficients in the database.

1. Access your UVM Zoofiles from your computer

Go to:

http://www.uvm.edu/artsandsciences/computingsvs/moving_data/?Page=zoofiles/zoofiles.html&SM=zoofiles/zoosub.html for instructions on how to access your UVM Zoofiles. After setting up your Zoofiles, create a folder within your main directory. Your main directory should be your UVM Net ID. In my case, my directory is named "aberger4" and I created a folder called "weka".

2. Download Weka on your Zoo Account

In order to download Weka, go to: http://www.cs.waikato.ac.nz/ml/weka/downloading.html and select the Stable book 3rd edition version for Unix/Linux machines, "Other platforms (Linux, etc.)". Initially, download the Weka folder (called "weka-3-6-10" in my case, though the name may change when a new version of Weka is released) locally to your computer's hard drive (just make sure you know where the folder is stored). Next, navigate to where you stored the "weka-3-6-10" folder and copy and paste the folder into the "weka" folder that you previously created in your zoo account.

3. Create a Java program that interacts with Weka

Since Weka was created in Java, you also will need to interact with Weka in Java. Weka provides Java packages that can be imported to interact with its functions. Be sure to store your Java program in that "weka" folder on your zoo account. After the Java program is created, it will need to be automatically run on your zoo account. Therefore, you will need to compile and run the Java program from the command line (not the IDE), since the version of java on your machine will likely be different than what is on your zoo account. For this reason, I recommend using a lightweight IDE such as jGRASP, which does not create a complex network of folders in your directory for every java project. Using more sophisticated IDEs, such as Eclipse and NetBeans, will still work but increase the complexity of the commands. In your Java program, be sure to import the necessary weka packages. In my case, I needed to add the following lines of code:

```
import weka.classifiers.functions.LinearRegression;
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSource;
```

When you read and write from external files in your java program, be sure to identify them by their full path name. In my case, the full path was: **/users/a/b/aberger4/weka/**. Weka can only use certain types of files that are in specific formats. Some good file types include ARFF and CSV. After you have created your java program, you will need to compile the program from the command line. So, open up a command prompt for your zoo account and navigate to your java program directory. In order to compile correctly, you will need to

reference the weka.jar file in the weka-3-6-10 folder, so it is important that the weka-3-6-10 folder is in the same directory as your java file. At the command line, type the following command:

javac –cp .:weka-3-6-10/weka.jar WekaTest.java

This command assumes that **weka-3-6-10** is the name of your downloaded weka folder and **WekaTest.java** is the name of your java program. If these names are different, then change the command to reference the correct names of your files/folders. Also, if you used a sophisticated IDE such as NetBeans, this command will be more complex since the files will be stored in subfolders. If the program is compiled correctly, then a **WekaTest.class** file should be created. If you have complier errors, then you will need to modify your java code and try compiling again. Once the program has been compiled successfully, you will need to create a jar file that can execute the program from the command line. When a jar file is created, a file called MANIFEST.MF is also created that is used to run the program. However, you will need to add some lines to this file in order for the jar file to execute properly. In order to add some lines, open a basic text editor (such as Notepad) and save the file as "manifest.txt" in the same folder on your zoo account as your java program. Now, add the following lines to your manifest.txt file:

Main-Class: WekaTest

Class-Path: weka.jar

If you named your java file something different than WekaTest, then alter these lines so that the Main-Class attribute matches the name of your file. After the manifest.txt file has been created, you can create the jar file with the following command:

jar cfm project.jar manifest.txt WekaTest.class

This command assumes that WekaTest.class is the name of your class file and that you would like to name your jar file project.jar. If the jar file is successfully created, you should be able to see the project.jar file in your directory after executing this command. In order to test the jar file to make sure it works, use the following command:

java –jar project.jar

If the jar file executes properly, then you are ready to start creating the php file.

4. **Create a php file that executes the java jar file**

The php file will need to retrieve all data (user responses) from the database and write them to an external ARFF or CSV file in a format that Weka can understand. However, since the responses to some questions will be null or missing entirely, it is important to be very careful when processing the data. All null (skipped questions) and missing (questions not seen if user exits survey early) responses will need to be replaced with the mean response for the corresponding question before the responses are written to an external file. Therefore, you will need to calculate the average response for every question and use these averages to

replace missing data so that there is a numerical response for every question. After all the data has been modified and written to an external file, you can execute the java jar file with php's shell_exec function that returns the java output, which is the coefficient for every question (variable). The coefficients are returned as a string separated by a space. Then you will need to store each coefficient in tblQuestion with their corresponding question and store the constant (y-intercept) in tblConstant. Once the php file is complete, you can execute the file at the command line with the following command:

php weka.php

assuming that weka.php is the name of your php file and that you are in the correct directory.

The following code on this php page needs to be modified for other projects:

- **Lines 7 and 8:**

  $dbUsername = "aberger4";
  $dbUserPass = "aideivim";

  These variables represent the database's username and password. Replace the values of these variables with your own database's username and password.

- **Line 46, 281, 322**

  mysql_select_db("ABERGER4", $connectID);

  "ABERGER4" is the name of my database. Replace this value with the name of your database.

- **Line 73, 77, 111, 241, 257, 259**

  $userSavings

Change the name of this variable to match the name of your response variable.

- **Line 221**

  fwrite($file, "Savings, ");

  Change “Savings” to match your desired response variable.

- **Lines 239, 241, 242, 257, 259, 260**

  $savingsIndex

  Change the name of this variable to match your desired response variable.

- **Line 272**

  $javaOutput = shell_exec('java -jar project.jar');

  Change “project.jar” to match the name of your java jar file.

III. Setting up the Cron Job

Cron jobs can easily be created on your zoo account from the command line. Open a command prompt, and type: **crontab –e**.

This command will open up a place where you can write the cron job command. The cron job command will consist of 5 stars (will indicate how frequently the cron job will run) and a path to the file to execute. So, to run a php file on your zoo account, your command will look something like this:

*** * * * * php /users/a/b/aberger4/weka/weka.php**

This cron job will execute the php file every minute. If we change the command to:

0 * * * * php /users/a/b/aberger4/weka/weka.php

The cron job will now execute the php file every hour (0 minutes past every hour)