JTEC User Manual

Calibration

The calibration of our software was built to allow the system to acclimate to the user's environment and facial features. This is done by asking the user to click on various points on the screen while looking pointedly at those points. This adds data points to WebGazer's regression model for more accurate predictions. It is a slightly transparent overlay that covers the normal webpage without impeding the webpage's loading. The center of the screen is dedicated to instructions so that the user of the website can properly complete the calibration.

The calibration screen has a close button in the top right of the window so that the user can close the calibration screen and subsequently opt out of the eye tracking software. The user will first be prompted to make sure they want to opt out of the eye tracking software before the software is stopped. In the bottom right of the window there is a help button; clicking this button will open a prompt displaying the calibration instructions in case the user needs to look back at them.

Once calibration is complete the overlay will disappear and the user can continue normal use of the webpage until the first validation period begins.

Validation

The validation is built using the same type of overlay as the calibration. After a set time as declared by the website developer, the validation overlay will appear. The user is prompted to look at and click each of the red dots until they disappear. After doing that for each quadrant of the screen, the data packet from that validation period will be sent to the server and the overlay will disappear and the user will continue the website until the next validation.

During the validation period, the overlay appears and waits until the user clicks on a dot. On the event of a click of a red dot, the software checks whether the prediction point from webgazer is within a certain distance from the location of the red dot. It does this for three seconds and then the red dot will move and wait for the user to click again.

In between validation period the software just appends a list of the x-points, y-points and time points for each prediction. At the end of a validation period we make a call to a function that will create a packet to send to the server with all the data collected. It also calculates a percent accuracy by dividing the number of points that were correctly predicted by the total points during validation.

_

Running the Parse Dashboard

Running the parse dashboard is very simple. Simply, using the terminal, go into the directory that contains the dashboard_config.json file (BackEnd directory). Then, in the terminal, enter the line:

parse-dashboard --config [name of dashboard config file]

For the example below, the dashboard config file is named dashboard-config-AWS.json, so the command "parse-dashboard --config dashboard-config-AWS.json" is entered into the terminal (see below).

```
wifi179-108:BackEnd admin$ parse-dashboard --config dashboard-config-AWS.json
```

This will generate the line:

```
The dashboard is now available at http://0.0.0.0:4040/
```

The dashboard can now be accessed with a web browser at the address: "localhost/4040"

Once on the dashboard, create a class called "EyeData." When a query is generated, the attributes of the class will populate themselves.

Querying and Generating Heat Maps

In able to get data from the database it is necessary to run a query for the specific data. There are two ways in which the data can be queried. Either the most recent data can be queried, or a specific data set can be queried based on the userID.

When performing a query, there are certain parameters to customize to your needs. The Query.html file in the Analytics folder contains all of the information for querying.

To perform a query on a specific data set:

1) Find the UserID in the dashboard
2) In Query.html, uncomment the line with the "generate_query" command and comment out the "get_most_recent_user" command (see below)

```
generate_query("ZMb6idm8If", generate_heatmap,0); // use for specific user

//get_most_recent_user(recent_user_heatmap);  //use for most recent user
```

3) Edit the 1st and 3rd parameters according to what dataset you want, and the validi_threshold desired. The picture above is performing a query for the userID "ZMb6idm8If" with a valid_threshold of 0 (meaning if the data has valid_threshold greater than 0 it will be displayed).
4) To change the binning size of the heat map, in the file "Plotly.js" edit the last parameter of the following line in the function make_heatmap()

```
  z: populate_array_binned(x_array, y_array,screen_size,25),
```

The line above creates a heat map with 25x25 pixel blocks. It can be made to as small as 1, where each point of the heat map represents a single pixel, and as large a number as you desire.

6) Lastly, double click the file "Query.html" to run the query.

To perform a query on a specific data set:

1) In Query.html, uncomment the line with the “get_most_recent_user” command and comment out the “generate_query” command (see photo below)

```
//generate_query("ZMb6idm8If", generate_heatmap,0); // use for specific user

get_most_recent_user(recent_user_heatmap);  //use for most recent user
```

2) The valid_threshold can be changed in the function “recent_user_heatmap()”. In the photo below the valid_threshold is set to 0

```
function recent_user_heatmap(user_var){
    generate_query(user_var.id,generate_heatmap,0)
}
```

3) Follow steps 5 and 6 above to finish generating the query.