Dimensional Personality Disorder Assessment

2017-12-19 BY GENE

Timothy W. Butcher and I collaborated on a number of diverse projects. One was a personality disorder quiz that he devised while taking an abnormal psychology university course.

Since 1999, when we made it, the quiz and the anonymous data and chart history has been lost to time. But recently I found and resurrected the quiz itself!

It is **196** questions long – Whew! But that *is* what it takes for an accurate questionnaire about multiple personality disorders.

Unfortunately, a number of the questions are difficult in their wording. I wish Tim was still around to discuss, clarify, and refine them…

~

Back then, my part was to make it interactive and "on-line." Briefly, here is how it works, underneath the hood. (And here is the code.)

For a user interface this time around, I chose Dancer2 as a web framework. Also, I need to chart the results and for that I chose to use the GD::Graph::Data module:

```
package DPDA;

use Dancer2;
use GD::Graph::bars;
```

There are a few routes ("web addresses that are understood by the app"), but the most important ones are for question/response and final result computation.

To keep track of the question/answer history for a quiz taker, I use a simple cookie. This allows multiple questionnaires to be running concurrently. Also, this allows a person to take a break from the quiz (as long as they don't clear their browser cookies).

For the question route, I first get our quiz progress and then handle the history. This is done by clearing the history **if** we are on question number one, and to transform it into an array and hash reference for later use, otherwise.

```
get '/question' => sub {
    my $progress = query_parameters->get('progress') || 1;

    # Handle the history
    my ( $history, %history );
    # Clear the history if on the first question
    if ( $progress == 1 ) {
        cookie( history => '' )
    }
    else {
        # Turn the history into an arrayref and a hash
        $history = cookie('history');
        if ( $history ) {
            $history = [ split /,/, $history ];
            %history = map { split /\|/, $_ } @$history;
        }
    }
```

Next we load the quiz questions and get a random, unseen one:

```
    # Load the quiz questions
    my @quiz = _load_quiz();

    # Get a random question that has not been seen
    my ( $question_num, $question_text ) = _get_question( \@quiz, \%history
```

The _load_quiz() subroutine does what you would expect – It just reads the question file, line by line, and returns the quiz as an array – one line per question.

The _get_question() subroutine is pretty simple too – It grabs an unseen question with this logic:

```
sub _get_question {
    my ( $questions, $history ) = @_;
    ...
    # Get a question that has not been seen
    my $question_num;
    do {
        $question_num = int rand @$questions;
    }
    while exists $history->{$question_num};
```

Ok. When a response has been made to a question, by submitting on the web, the quiz route is invoked:

```
post '/quiz' => sub {
    my $question = body_parameters->get('question');
    my $answer   = body_parameters->get('answer');
    my $progress = body_parameters->get('progress');
```

Next, a “question|answer” item added to the history, and the progress is incremented. If we are past the last question, then go to the finalize chart route, otherwise, ask another question!

```
    if ( $progress > @quiz ) {
        redirect '/chart';
    }
    else {
```

```
        redirect '/question?progress=' . $progress;
    }
```

And what is that last step? Well, I defer to the code itself. Again, if Tim were here, I would ask him for the statistical formulae being used, so that I could say succinctly what is being computed: In the final "/chart" route, the history is handled as above, and the quiz is loaded to access the metadata for each question – "category" and "polarity." Next, the results are calculated, the categories ordered, the average responses calculated, the "discord" ("dishonesty", "ambivalence") calculated and finally a bar chart is rendered.

Now to test that the quiz results were accurate, I wrote a little Test::WWW::Mechanize script to test the boundaries. You can see the results of having every disorder to having none of the disorders, on the quiz Sample Results page.

FILED UNDER: EDUCATION, INTERNET
TAGGED WITH: PERL, PSYCHOLOGY, SURVEY, TIM

Epistemologist-at-large

^ Top