Iterated Drawing Application

Web application for Professor De Boer @ VUB AI Lab, concerning language evolution. The Django framework has been used to implement, following a model-view-controller model. Verification and storage of data is handled by the backend part of the project.

The source code of the project is contained within the *src* folder.

Setup

In the *src* folder, the *activate_experiment.sh* file can be executed to start the Django server (in developer modus). **Make sure variables in Experiment.py are correct before setting up the project. These are mainly paths to directories like "cases" and "experiments" in order to load the sessions.**

1. Open terminal in project directory and run the *activate_experiment.sh* script. This will create a virtual python runtime environment for the application to run, installed with the necessary modules.
2. The initial cases are loaded into the experiment (form the *data/cases* folder). This can be changed in the *Experiment.py* when other cases have to be loaded for the first time.
3. Go to 127.0.0.1:8000 in the browser for interaction on the local machine. For local network access, open the browser on the client machine (tablet or phone) and go to the IP address of the host (using the command *ip addr* to find the IP of the host), including port number 8000 (make sure the machine's fire wall doesn't block the outgoing data).
4. After an experiment is completed, the data is saved as described in the **Data** section.
5. To close the project: press ctrl+c in the console.

At the first session by default, all cases are considered in the chosen directory at random. This can be specified in the load_cases method of the Experiment class by using a list of strings containing the names of the actions to test.

New cases can be added to the case directory as CSV files. When added, the name of the action is added to the *actions.json* file located in the same directory. This ensures the correct mapping of the action name to the file name containing the data of the case action.

Tolerance for the training session cases can be changed int the TrainingSession class. This is the distance of error allowed for a training case when verified by the server.

Frontend

The statics folder contains all static elements like JavaScript, CSS and images. The *canvas.js* file contains the draw application as well as path recording functions during the Session.

Backend

The backend uses several abstractions in order to execute an experiment. An **Experiment** class consists of a **TrainingSession** and a **TestingSession** class. Experiment is responsible for the correct loading of case.

A Session consists of several Cases that could be loaded into the experiment. Each Case also consists of Trial classes who're responsible for evaluating the input of every user trial.

Form

In *src/form//models.py*, the form can be defined for registering a new user. Default, first/second name, mail and age are asked.

RESTful API

A simple RESTful API is used for data transfer between client and host while on a page. For both kind of Session there is a **get_case** and a **post_case** method. JSON objects and strings are used as the transfered data.

Call	Arguments	Response
training/get_case	none	JSON: {action:String, path:Array{x:Int, y:Int, t:Int}}
training/post_case	JSON: Array{x:Int, y:Int, t:Int}	String:"tolerated", "not tolerated", "session done"
testing/get_case	none	JSON: {action:String, path:Array{x:Int, y:Int, t:Int}}
testing/post_case	JSON: Array{x:Int, y:Int, t:Int}	String: "tolerated", "session done"

String indicate the status of a session. There are 3 possible values:

- **tolerated**: The sent path is tolerated by the system using according to the tolerance value. The next case is prepared when a *get_case* is called.
- **not tolerated**: Sent input is not accurate enough or drawn in the wrong direction. The user has to retry the case and the current case is returned when *get_case* is called.
- **session done**: All cases loaded into the session are traversed and he session is done. Go on to the next phase of the experiment, which is testing or the end page.

Data

After the testing phase, the data of each user is saved by the backend in the *data/experiments* folder. The name of the directory is a hash of the user's name and the timestamp of the experiment. It contains following items:

- **info.json**: Contains the information of the user and the cases the user has traversed, both training and testing cases.
- **actions.json**: This file maps the names of the actions the user got during the testing phase onto the file names containing the recorded path data.
- **record files**: CSV files containing the path the user drew during the experiment. Containing of 3 columns:
    - X coordinate of record point
    - Y coordinate of record point
    - T time in milliseconds the user passed the record point