Mattia Amato, 10-919-686

Software Manual - Instructions

1. Basic Structure

The CrowdSA platform is divided in two different SBT projects: the client and the server. A copy of these two applications are available in the CD-ROM or in the following GitHub repositories:

- Client:
  - **https://github.com/Mattiamato/CrowdSA_client**

- Server:
  - **https://github.com/Mattiamato/CrowdSA_server**

Each of these applications contains an "application.conf" file which lists all the configurable parameters to connect to the database and to run the application correctly.

The server, once started, will try to connect to the database defined in the configuration file and, if needed, will ask to apply an evolution to it in order to create all the necessary tables.
A dump of the database's structure for the server application is available in the above-mentioned repositories or in the CD-ROM. The same also holds for the client application.

1.1 Server's structure

The server is a web application developed with the Play Framework. This MVC based application is partitioned in different packages:
- Models
  - In the models package there are the implementations of all the main objects used by the system. All the tables present in that database are defined in this package.

- Views
  - The view package contains all the HTML pages. The page which is dynamically completed with the other partial HTML pages is the "main.scala.html" page. This webpage includes the header bar, present in all the other pages, and a dynamic content field which is overwritten every time a new page is requested.

- Controllers
  - The controllers package contains all the objects which are called in order do execute some actions, e.g., the post method, defined in the Login controller, is called when a user wants to log into the system.

- Pdf
  - The pdf package includes the classes to highlight some text on a pdf file.

- Persistence
  - The persistence package contains all the objects which directly communicate with the database.

Another important folder in the server's structure is the public directory. This directory contains some JavaScript files used to load the PDFjs viewer. An extra directory is present for the AngularJS part of the system. AngularJS allows to integrate some highly dynamic objects on the pages and is used in several occasions to GET or POST some information to the server.

The pdfs directory is the directory where all the papers loaded into the server are stored. This directory contains some initial pdfs which can be used to test the system.

1.2 Client's structure

The client is a Scala application based on PPLib and allows to communicate to the server which paper a user wants to analyse. The client contains all the processes which automatically create and execute all the possible variants at the same time. These processes are listed in the recombination class.
This application is divided in different packages and their structure is similar to the one used in PPLib.

- Crowdsa
  - This package contains all the elements which are used by PPLib to create a new portal, process the queries and send some basic commands to the server, e.g., to upload a question or to get all the assignments created for a specific turker.
  - The CrowdSAQuery class is a central element for the client. This class is used as a container for the queries and the properties of each request.

- Patterns
  - The patterns package includes the executor of the Iterative Refinement process which was disabled for the experiment executed in the thesis.

- Process
  - The process package is similar to the one of PPLib. It contains all the processes which can be used in the PPLib Process Repository (PPR). In this package the Discovery Process as well as the Assumption Process are defined. Another important class is "ExtractStatisticProcess": it is the entry point used to execute first the discovery step, then the assumption step.
  - The recombinations listed in the "ExtractStatisticsRecombination" object are used by the client and initialized at runtime. For the thesis it was requested only the Collect-Decide process. The other processes are commented out.

The entry point of the application is the Main.scala object. Two parameters are needed to run the application: first, the PDF file and second, its title which will be displayed on the server.

Since the client relies on PPLib, it is necessary to download this library and compile it. PPLib was integrated in the client application the 20th of April, 2015. Next or previous versions may not be compatible with the platform.

2. Usage

In order to be able to use correctly the client application the server needs to be running and the evolution to be applied to the database.

To start the server, first of all the "application.conf" file needs to be updated with the according parameters (database location, username, password, …), then the server can be started as a normal SBT project through a shell with the following command:

- **java -jar sbt-lanuch.jar run**

The sbt-launcher.jar is the SBT application.

Once the server is running the client can be started. The client, as perviously mentioned, needs two different parameters: the path to the PDF file the user wants to analyse and its title. After changing the parameters of the “application.conf” file, where the details of the connection to the database are defined as well as the address of the server, the client can be started with, e.g., the following command:

- **java -jar sbt-launch.jar "run ../publication.pdf \”Title of the publication\""**

The path to the PDF file is in this example “../publication.pdf” and its title is “Title of the publication”.

Once the client is running all the missing tables are created and, in some cases filled in with data, in the database as well as the questions which are automatically sent to the server.

As soon as the tables containing predefined values are created, as in the case of the statistical methods which can be matched to a PDF, they are not going to change anymore. This is also valid if we start the application several times without changing the target database.

At this point, after logging into the web application, the user will be able to view the paper and the questions generated by the client.

Once an answer is generated in the server, the client will get and store it into its database. When all the questions are answered, an evaluation of the paper is performed. This evaluation is part of the ExtractStatisticProcess in the process package.

The results, together with the overall cost and the time needed to evaluate the paper, are stored in the client’s database in the Discovery table.