Mattia Amato, 10-919-686

Software Manual

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 “application.conf” file and, if needed, will ask to apply an evolution to it in order to create all the tables needed.
A dump of the database’s structure for the server application is to find 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 application is subdivided 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 contains 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., when a user wants to log into the system the post method , defined in the Login controller, is called.

- Pdf
  - The pdf package contains 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. This application allows to communicate to the server which paper a user want to analyse. The client manages the processes and based on the recombination class, creates automatically all the possible variants and execute them at the same time. This application is subdivided in different packages. The structure of these packages is similar to the one of PPLib.

- hcomp.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 create 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 contains 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). Here it is defined the Discovery Process as well as the Assumption Process. Another important class is "ExtractStatisticProcess". This class is the entry point used to execute, first the discovery step, then the assumption step.
  - The Recombination used by the client are listed in the "ExtractStatisticsRecombination" object. This object contains a list of recombinations which have to be created 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. To run the application two parameters are needed to be passed as arguments to the client: first the pdf file and second the title of the pdf 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 correctly use the client application the server needs to be running and the evolution needs 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 trough a shell with the following command:

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

Where the sbt-launcher.jar is the executable of the SBT application.

Once the server is running the client can be started. The client as above-mentioned needs two different parameters: the path to the PDF file the user want to analyse and the title of it. 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\""**

Where the path to the PDF file is “../publication.pdf” and the title of the document is “Title of the publication”.

Once the client is running it will create all the tables in its database as well as the questions which are automatically sent to the server.

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, then an evaluation of the paper is performed. This evaluation is part of the ExtractStatisticProcess under 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.