Exercise PC Systems Development & Infrastructure JKU Linz

2012W, Exercise 1 Institut für Pervasive Computing

Return till: Wed, 24. Oct 2012, 12:00 Gerold Hölzl, Michael Matscheko

Before you can begin with this exercise, set up your Android Development Environment with Eclipse. Check the slides for step-by-step instructions or follow the guide at http://developer.android.com/sdk/installing.htm.

1) Recording ARFF Data Sets (20 points)

Weka (Waikato Environment for Knowledge Analysis) is a popular machine learning suite, which we will use in the following exercises for analysis and classification of data. Weka loads attributed data sets from ARFF (Attribute-Relation File Format) files. The ARFF file format is rather simple: in an ASCII text file it first declares the attributes and then lists the data line by line. Below is an example with acceleration data:

```
% Group: groupname
% Date: 2012-10-04 13:36

@RELATION systemsdevelopment

@ATTRIBUTE timestamp      NUMERIC
@ATTRIBUTE accelerationx  NUMERIC
@ATTRIBUTE accelerationy  NUMERIC
@ATTRIBUTE accelerationz  NUMERIC

@DATA
1331074800000,5.1,3.5,1.4
1331074800002,5.4,3.3,1.5
1331074800005,5.3,3.3,1.5
1331074800007,5.6,3.2,1.4
1331074800010,5.7,3.1,1.3
1331074800012,5.4,3.3,1.3
1331074800015,5.5,3.2,1.4
1331074800017,5.8,3.0,1.3
1331074800020,5.7,3.1,1.3
1331074800022,5.9,3.0,1.4
1331074800025,5.9,3.0,1.4
```

Comment Block: Always include the names of your group members and the date when the file was recorded in the comments at the beginning of the file.

Relation: The relation is irrelevant in this exercise, use "systemsdevelopment".

Attributes: Here we declare all the attributes we have in our data set. We will record 3-axis acceleration data on our phone and need a timestamp (ms since 1970) and 3 acceleration values (m/s^2).

Data Segment: The data segment contains each recorded instance on a single line. Attributes are separated by comma and must appear in the same order as declared above.

A more extensive description of the ARFF file format can be found here:
http://weka.wikispaces.com/ARFF+%28stable+version%29

In this exercise you have to write a recorder application for the accelerometer in an Android smartphone. It will output the data to an ARFF file as described above.

Name your new project `ARFFRecorder`. The application requires 2 components: A service that can stay in the background and does the actual work, and an activity that implements a simple GUI to control the service.

ARFFRecorder Activity

Eclipse will automatically create most files you need.

- *res/layout/main.xml*: Change the GUI so that you have at least a button to start and stop the recorder service. You may also add widgets to change the filename, etc. (optional).
- *src/…/ARFFRecorderActivity.java*: Should already contain onCreate and the code to display the main layout. Add methods to start and stop the service.

Image 1: Sample GUI.

Your GUI does not have to look like this!

Design something awesome ;)

ARFFRecorder Service

The service has to show a notification in the system notification area (top of the screen) for as long as it is recording.

You can use the *Local Service Sample* in the Android Reference as a starting point:
http://developer.android.com/reference/android/app/Service.html

The sample already contains all the code you need to set up the service, display notifications and interact with the activity (by binding to it). Remember that you also need to start the service with `startService()` otherwise it will terminate with the activity. Binding to it and starting it will not produce a second service instance; there is always a single instance for all clients.

Implement methods to collect samples from the accelerometer sensor (slides contain an example), and write them to an ARFF file. When registering as a Listener to SensorEvents, you can choose between different data rates. Record at the highest possible rate: `SensorManager.SENSOR_DELAY_FASTEST`.

Warning: The timestamp you get from the sensor (`SensorEvent.timestamp`) specifies nanoseconds since system startup and not an absolute date/time. Either use `System.currentTimeMillis()` to get an absolute timestamp in ms, or convert the sensor event timestamp like this:

```
// somewhere during initialization
long timeOffsetMs = System.currentTimeMillis() - System.nanoTime() / 1000000;

// in featuresChanged(FeaturesEvent f)
long actualTimeMs = timeOffsetMs + f.endTime / 1000000;
```

Guidelines:

- Zip your eclipse project folder and upload it.
- Your source code has to include comments where appropriate:
  - Before methods explain in 1 to 2 sentences what the method does, unless it is a simple utility method and it is obvious anyway.
  - Inside methods, add short comments (only a few words) before semantically separate statement blocks.
- Add a `readme` file! It should contain your group members, a summary of what you did (just a few sentences) and any problems you had.
- We have a forum at https://www.pervasive.jku.at/Forum/. If you need help, use it!