SSI/ModelUI Tutorial

v.0.9
compiled: October 16, 2013

Multimedia Concepts and their Applications

Johannes Wagner

http://openssi.net
wagner@openssi.net

SSI/ModelUI is a graphical interface that allows users to record, evaluate and train own recognition models. Trained models are ready to be used in real-time in a SSI pipeline. The interface covers five main tasks:

1. Organize multiple projects.
2. Add new recordings to a project.
3. View recordings and create/adjust according annotations.
4. Automatic feature extraction and model training.
5. Test a trained model in real-time.

Contents

Chapter 1

The Project Panel

The project panel helps the user to organize the recordings made within the different projects. On disk a project has the following fixed folder structure:

```
> root       // name of project
  |- data    // all data recorded within the project
    |- userA // recorded data of a single user
      |- 2010-12-06_14-46-12 // data of a single session
      |- 2010-12-06_14-51-34
    |- userB
    |- userX
  |- eval    // folders with evaluation results
    |- 2010-12-02_14-37-16
    |- 2010-12-02_14-42-45
  |- log     // log files
  |- opts    // option files
  |- stimuli // stimuli files
  |- train   // folders with trained models
    |- 2010-12-02_14-37-16
    |- 2010-12-02_14-42-45
```

Since, the project panel automatically manages the folder and file structure of a project, it is usually not necessary to directly manipulate the file structure of a project.

1.1 Overview

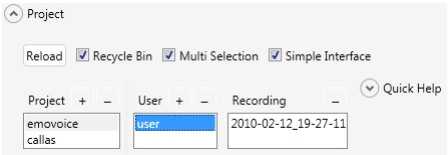

Figure 1.1: The project panel.

The project panel is displayed on top and always visible (unless the user hides it by clicking the expander button on the top left). It is used to control the active project and browse recordings. Actions in the other panels often depend on the current selection.

The list on the most left shows a collection of all projects. A project holds recordings of one or more users and the trained models. Selecting a project from the list will load it. Users within the project are now displayed. The `+` button on the top of the list can be used to add new users. When a user name is selected according recordings are loaded. By holding down the ctrl-button and selecting several names it is possible to load recordings of multiple users (`Multi Selection` must be checked).

The `-` button above each list allows the user to delete selected items. In case of projects this will only cause the project to be excluded from the list of projects. For users and dates all related items (i.e. signal, annotation and sample files) will be deleted from disk. However, if the check box `Recycle Bin` is checked files will be moved to the recycle bin and can be recovered. The `Reload` button can be used to refresh the currently selected project.

1.2 Creating a Project

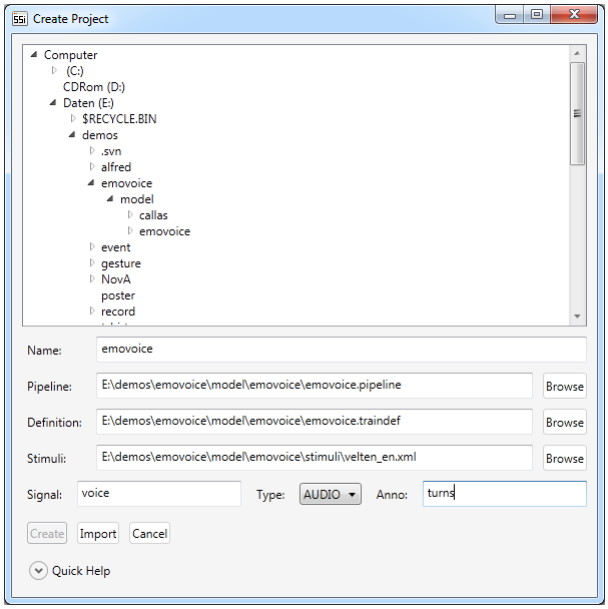

Figure 1.2: Dialogue for creating new or importing existing projects.

The `+` button above the list of projects allows the user to either create a new project or import an existing project. A dialogue allows the user to enter the project settings.

First, the user must select a root folder on disk using the folder dialogue on top. Selecting a folder, which already contains a project will automatically activate the `'Import'` button. In this case now further settings are required. To create a new project enter a project name. A folder with this name will be automatically created. All recorded signals, annotations, samples and models, as well as, stimuli files will be created inside this folder. To move a project to another computer it is sufficient to copy the whole project folder and import it.

Next, a pipeline is set (ending to `*.pipeline`). Apart from the sensor and possible pre-processing steps, the pipeline must include a component `'ssi_consumer_MlpXml'`. It takes two signals as input: the signal we apply recognition to and an additional input (`xinput`), which is the trigger signal. The trigger signal indicates when a recognition event occurs. If the current sample value is non-zero recognition starts. When sample

values become zero again recognition stops. E.g. in case of a speech recognizer the trigger signal would be the results of a voice activity detection (VAD). Though the component has several options, they will be automatically set except for the `type` option, which determines the storage format (0=ssi stream, 1=audio wav file, 2=video avi file). Here is an example for an audio signal triggered with a voice activity detection stored as an audio wav file:

```
<consumer create="ssi_consumer_MlpXml" type="1">
   <input pin="audio" frame="0.2s" delta="0"/>
   <xinput size="1">
      <input pin="audio_vad"/>
   </xinput>
</consumer>
```

Now, a training definition file is selected (ending to `*.traindef`). It includes information on available feature extraction and classifier methods. Each `item` is given a unique name, which later on allows the user to choose between the different methods. Each item includes an optional `transform` and a mandatory `model` element. If a transformer is given it will be applied to the input stream before recognition in order to apply feature extraction. The following code carries on the above example and adds two classifiers, namely Naive Bayes and Support Vector Machines, together with a feature extraction component.

```
<?xml version="1.0" ?>
<traindef ssi-v="1">
   <item name="ev-bayes">
      <transform create="ssi_feature_AudioFeatures"/>
      <model create="ssi_model_NaiveBayes"/>
   </item>
   <item name="ev-svm">
      <transform create="ssi_feature_AudioFeatures"/>
      <model create="ssi_model_SVM"/>
   </item>
</traindef>
```

In addition, the user has the option to select a stimuli file. A stimuli file stores slides displayed during a recording. They usually include some instruction or stimuli and ask the user to perform certain actions or try to elicit certain behaviour. Each slide consists of a source, a trigger and a label. The source defines the content of the slide, which will be displayed as a html file in the web browser. The trigger defines how or when the transition to the next slide will happen. And the label is a description of the action or desired user state. Stimuli files follow a certain xml structure (see Table 1.1 and 1.2 for available definitions):

```
<stimuli>
  <list size="x"> // number of slides in the stimuli
    <slide label="xxxxx"> // a single slide
        <source id="x" .../> // what is displayed
        <trigger id="x" .../> // how long is it displayed
    </slide>
    ...
```

id	description	example
Text	plain text	<source id="Text" text="do this and that.." color="00FF00" size="5"/>
Code	html code	<source id="Code" code=" do something else.. " />
URL	linked page	<source id="URL" url="http://hcm-lab.de" />

Table 1.1: Supported definitions for stimuli sources.

id	description	example
Button	waits for user to press button	<trigger id="Button" />
Timer	waits x seconds	<trigger id="Timer" seconds="3"/>
Event	waits for x user events	<trigger id="Event" number="3"/>

Table 1.2: Supported definitions for stimuli trigger.

```
  </list>                                                    8
</stimuli>                                                   9
```

Finally, the type of the signal has to be selected. It has to correspond to the one used in the pipeline, i.e. stream, audio or video. Name of signal and annotation can be chosen arbitrarily.

When the user presses `Create` all files will be copied to the root folder and the project settings are stored in a file `'modelui.project.xml'` located in the root folder of the project. Afterwards the new project is added to the list of projects and loaded.

Chapter 2

The Record Panel

The record panel allows the user to make new recordings.

2.1 Overview

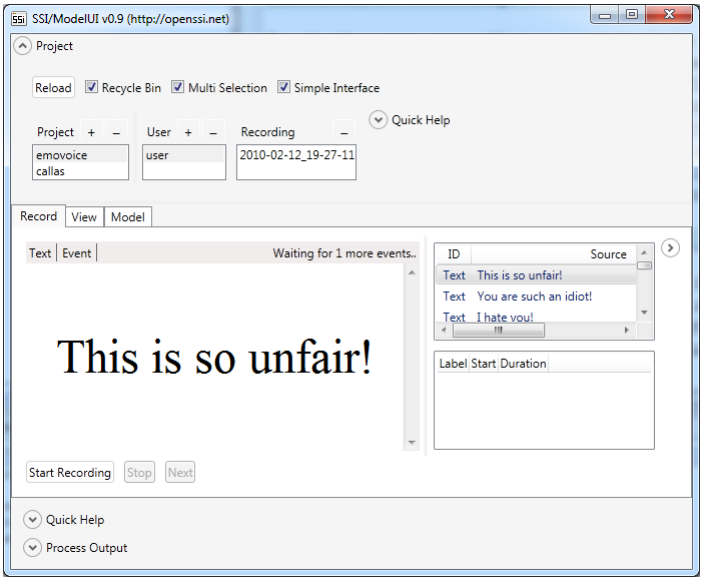

Figure 2.1: The record panel.

The record panel consists of a web browser to display the stimuli slides. If no recording is running the user can browse through the stimuli slides in the upper list on the right. If a slide is selected its content is displayed in the web browser. A status bar above the web browser gives additional information about the type of source and the kind of trigger connected to the slide (see Section 1.2). During a recording events triggered by the pipeline are collected in the lower list on the right.

2.2 New Recording

To start a new recording a user must be selected (if multiple user names are selected the first user in the list will be taken). A new recording is started by clicking on the `'Start Recording'` button below the web browser. As soon as the pipeline starts to record(which may take a couple of seconds) the first slide will be displayed. While the

user follows the instructions on the screen the pipeline continues to capture data and if an event (e.g. a utterance) is detected it will be displayed in the event list. Sometimes a certain number of events must be detected before the next slide is displayed. In other cases a countdown is used or the user is asked to press the `'Next'` button below the browser. The recording stops automatically when the last stimuli slide has been shown or the user presses the `'Stop'` button. Now, the pipeline will be stopped (which may again take a couple of seconds) and the recording is stored for the current user.

Chapter 3

The View Panel

The view panel allows the user to review recordings and adjust/create annotations to describe observed user behaviour.

3.1 Overview

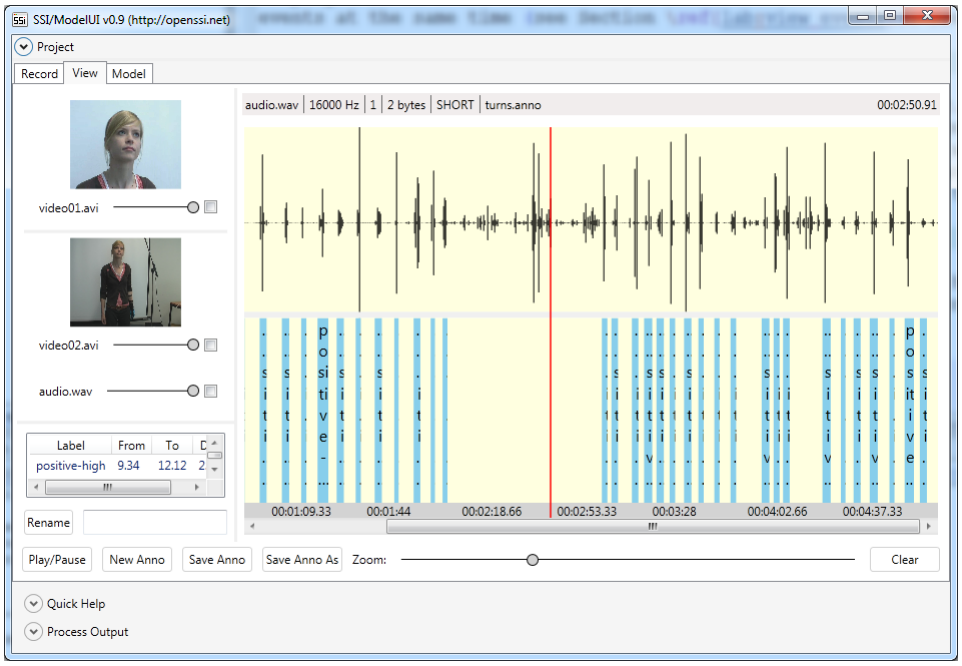

Figure 3.1: The view panel.

The view panel has a time line for navigation through the recording (see Section 3.2). To load the signals and annotations of a recording session double click the according recording. Videos are displayed on the left side together with a volume slider for each media. Events of the currently selected annotation track (yellow background) are listed in a table below the videos. It can be used to quickly jump to a certain event or change the label of several events at the same time (see Section 3.3). The `Clear` button removes all signals and annotations from the panel.

3.2 Navigation

The current position in the time line is marked by a red cursor. Clicking on a track moves the cursor to that position. If at least one media (audio or video) has been loaded, playback is possible by pressing the `space bar`. Usually, the recording is played until the end of the timeline unless an event has been selected. In this case the recording is looped within the borders of the events. An event is selected by clicking on it. As long as replay is active the user cannot change the position of the cursor. Pressing the `space bar` again stops replay. The slider on the navigation bar can be used to adjust the zoom level.

3.3 Annotations

Annotation files contain detected events. An event has a start and a stop time, as well as, a label string, which can be empty. An annotation file aggregates a set of events and is stored as a plain text file ending on `*.anno`, e.g.:

```
 8.9 13.5 labelA
16.0 19.6 labelB
21.6 24.8
27.2 30.7 labelA
```

Events of an annotation file are displayed as segment blocks on an annotation track. The label of an event is shown inside the box. new annotations can be created (`New`) and saved (`Save`, `Save As`). Right click on an annotation track adds a new event. By default the label is empty. Hitting the `enter key` allows the user to apply a string. Pressing the `del key` removes a selected event from the current annotation track. When the mouse is moved to the center of an event the cursor shape changes to a cross cursor. By holding the left mouse button pressed the event can now be moved along the track. Moving the mouse to the left or right border of an event allows the user to adjust its borders.

Chapter 4

The Model Panel

The model panel allows the user to evaluate and train models. The user can switch between both modes using the tab panel below the extraction bar.

4.1 Extraction

Figure 4.1: The extraction bar.

The extraction bar is used by both, the evaluation and the training panel to extract samples using the selected signal and annotation. The extraction methods defines the type of features that will be extracted from the signal and the type of model, which is trained from the extracted samples (see Section **??**).

To speed up consecutive runs on the same recordings, extracted samples are stored on disk with the following naming convention: `<signal>.<annotation>.<method>.samples`. If it becomes necessary to re-extract all samples, e.g. because of changed annotations, the check box `re-extract features` should be checked.

4.2 Evaluation

After selecting at least one recording session and a training method press the `Start Evaluation` button invoke an evaluation on the selected recording session(s). The following evaluation strategies are available:

1. full = the full training corpus will be used for evaluation
2. k-fold = the training corpus is split in k folds and each fold is left out once while the others are used to train the model
3. leave-one-user-out = one user is completely left out and the others are used for training, which is is repeated for each user
4. leave-one-sample-out = one sample is completely left out and the others are used for training, which is is repeated for each sample

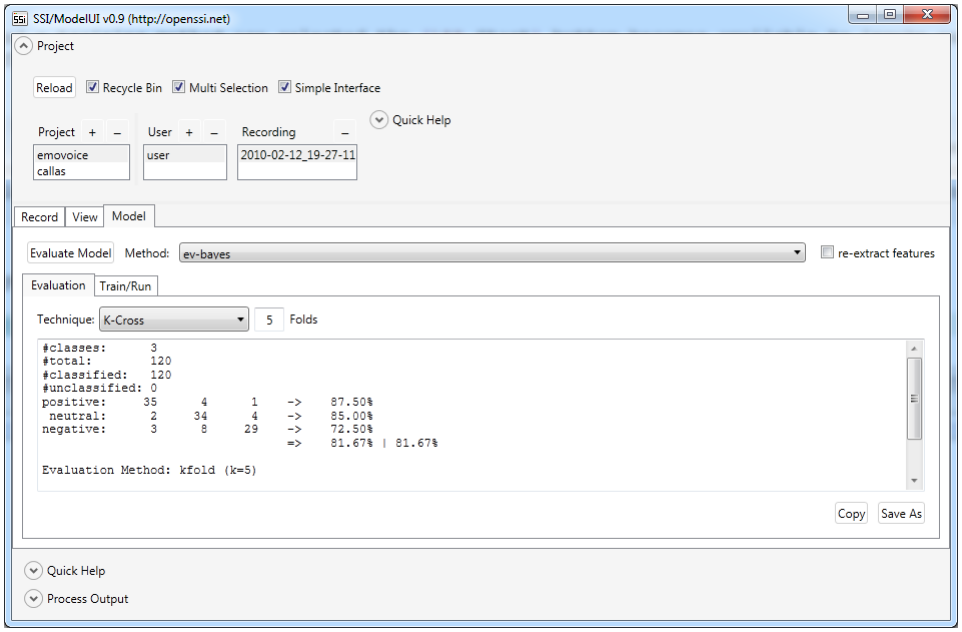

Figure 4.2: The evaluation panel.

The result of an evaluation is displayed as a confusion matrix and can be copied to the clipboard or stored in a text file.

4.3 Training

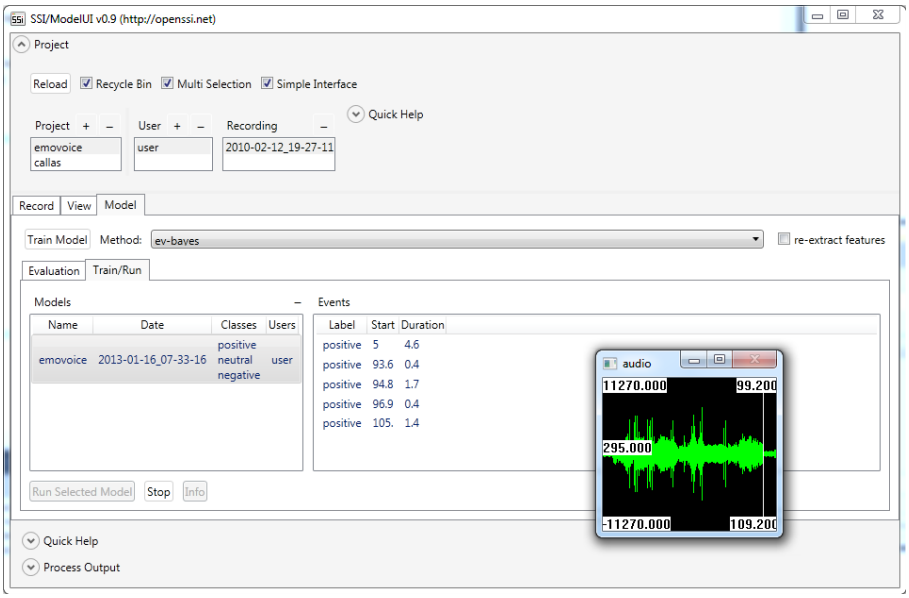

Figure 4.3: The training panel.

When the `Train/Run` mode is active and the `Train Model` button is pressed all selected recordings will be used to train a new model. When training is finished the new model is added to the list of available models. For each entry the class names and user names are displayed. Hitting the `Info` button opens a file with ending `training`. It

is generated during the training procedure and includes relevant information about the training, such as the name of the classifier or the included recordings:

```
<training ssi-v="1">
   <def>ev-bayes</def>
   <signal>audio</signal>
   <type>1</type>
   <anno>turns</anno>
   <paths>
      <item>2010-02-12_19-27-11</item>
   </paths>
</training>
```

Finally, a selected model can be directly tested by pressing `Run Selected Model`. Detected events are classified in real-time and displayed.

4.4 Using a Model

Models can be used outside the GUI in SSI pipelines if you make sure to apply as input the same (possibly pre-processed) signal. Let us pick up the previous example that was using an audio signal and a voice activity detection (VAD) to trigger the recognition component. After putting the audio sensor and a transformer to apply VAD in place, we need two more components to complete the pipeline. A `ZeroEventSender`, which picks up the VAD signal and sends an event if a non-zero segment of certain length (`mindur`) is found, and a `Classifier`, which takes the audio signal and listens to the trigger (here the event address has been set to `speech@audio`):

```
<consumer create="ssi_consumer_ZeroEventSender" mindur="1.0" hangin="3"
   hangout="3" sname="audio" ename="speech">
   <input pin="audio_vad" frame="0.1s"/>
</consumer>

<consumer create="ssi_consumer_Classifier" trainer="emovoice">
  <input pin="audio" listen="speech@audio">
     <transformer create="ssi_feature_EmoVoiceFeat"/>
  </input>
</consumer>
```

Note that it is not necessary to manually apply feature extraction or mention the classification method as these properties are stored along with the trained model, which consists of the following three files:

- `emovoice.trainer`
- `emovoice.ssi_model_NaiveBayes.model`
- `emovoice.00.ssi_feature_EmoVoiceFeat.option`

List of Figures

List of Tables