Deliverable D5.6
Second prototype of timbral characterisation tools for semantically annotating non-musical content

Grant agreement nr	688382
Project full title	Audio Commons: An Ecosystem for Creative Reuse of Audio Content
Project acronym	AudioCommons
Project duration	36 Months (February 2016 - January 2019)
Work package	WP5
Due date	31 July 2017 (M30)
Submission date	31 July 2017 (M30)
Report availability	Public (X), Confidential ( )
Deliverable type	Report ( ), Demonstrator (X), Other ( )
Task leader	Surrey
Authors	Andy Pearce, Saeid Safavi, Tim Brookes, Russell Mason, Wenwu Wang, and Mark Plumbley
Document status	Draft ( ), Final (x)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Table of contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

Executive Summary

This report describes the improvements and use of the demonstrator software for the eight perceptual models that can predict the timbral characteristics of a recorded sound by analysis of the audio file: *hardness*, *depth*, *brightness*, *roughness*, *warmth*, *sharpness*, *boominess*, and *reverb*. These models can be used to automatically generate metadata describing the timbral properties of recorded sounds, which can, in turn, be implemented into a search function, enabling users to filter search results based on the timbral properties.

Improvements have been made to the models of *hardness*, *depth*, and *brightness* since their initial prototypes described in Deliverable D5.2, adding new feature extraction algorithms and being calibrated on a much larger dataset of subjective ratings. Three new models are also included in this latest release that are yet to be evaluated: *warmth*, *sharpness*, and *boominess*. Evaluation of these models will be included in the future Deliverable D5.7.

From Deliverable D5.3, it was identified that the models of metallic-nature and reverb were extremely unreliable. Because of this, the model of metallic-nature has now been temporarily removed, and the model of reverb has been reworked entirely.

All developed perceptual models, with the exception of reverb, are implemented using the Python programming language. The new classification model of reverb has been developed and implemented with MATLAB and Weka. All current models have been made available in a public source code repository[1,2].

[1] https://github.com/AudioCommons/timbral_models
[2] https://github.com/saeidsafavi/timbral_models/tree/patch-1/Reverb/V1

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1 Description of the models

The Audio Commons project aims to provide tools for the automatic annotation of audio content. The development of these tools is spread across work packages (WP) 4 and 5, depending on the type of properties to be annotated. WP5 aims at providing annotation tools that describe non-musical properties; more specifically the timbral characteristics of sound effects.

In Deliverable D5.2, prototype timbral models of *hardness*, *depth*, *brightness*, *metallic_nature*, *reverb*, and *roughness* were described. The evaluation of these, in Deliverable D5.3, identified that the models of metallic-nature and reverb were extremely unreliable. Because of this, the model of metallic-nature has now been temporarily removed, and the model of reverb has been reworked entirely.

This demonstrator package describes eight perceptual models for annotating the timbral attributes of *hardness*, *depth*, *brightness*, *roughness*, *warmth*, *sharpness*, *boominess*, and *reverb*. All attributes, with the exception of the reverb, are coded in Python and are structured into a single `timbral_models` package. The reverb model was developed in MATLAB and Weka.

Section 1.1 below describes the installation, setup procedure, and dependencies required to run the timbral models. Sections 1.2 to 1.9 then describe each model in more detail, the required inputs, the outputs of the function, and provides some example code for running each model.

1.1 Model Installation

The developed timbre models are divided into two implementations: the *reverb* model in MATLAB and Weka; and all other models implemented in Python. This section describes the installation requirements for each implementation type.

1.1.1 MATLAB and Weka implementation

To obtain the perceived level of reverberation from audio files two separate platforms have been used:

- Matlab for feature extraction; and
- Weka for modelling and evaluations.

The feature extraction functions are available from the project's GitHub page: https://github.com/saeidsafavi/timbral_models/tree/patch-1/Reverb/V1

All the feature extraction functions are tested in Matlab R2017a. All the dependencies and sub-functions are included in the subfolder named "/src". For the feature extraction, there is no need to install extra packages or software. Only couple of parameters and file names need to be specified.

For the modelling and evaluation, the Weka toolbox is used. All the modelling and evaluation parts are carried out using the Weka version 3.8. This software can be downloaded from: https://www.cs.waikato.ac.nz/ml/weka/.

Weka is the Java based platform. All the requirements and documentations for installation of this toolbox can be accessed from https://www.cs.waikato.ac.nz/ml/weka/requirements.html and https://www.cs.waikato.ac.nz/ml/weka/documentation.html.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.1.2 Python implementation

The timbral models are available from the project's GitHub page: https://github.com/AudioCommons/timbral_models. All Python-implemented models were designed and tested in Python 2.7. These timbral models rely on the `numpy`, `scipy`, `soundfile`, `sklearn`, and `LibROSA` Python packages. These can all be all be installed using the pip tool, e.g. `pip install numpy`

Additionally, the code requires the use of the Essentia library. Documentation for the installation of Essentia can be found at http://essentia.upf.edu/documentation/installing.html.

All Python-based timbral models can be accessed by importing the `timbral_models` package into Python. At a minimum, each model requires a string that is the path and filename of an audio file. The general usage of all Python models is shown below.

```python
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
hardness = timbral_models.timbral_hardness(fname)
depth = timbral_models.timbral_depth(fname)
brightness = timbral_models.timbral_brightness(fname)
roughness = timbral_models.timbral_roughness(fname)
warmth = timbral_models.timbral_warmth(fname)
sharpness = timbral_models.timbral_sharpness(fname)
boominess = timbral_models.timbral_booming(fname)
```

All timbral models have at least two optional inputs of `dev_output` and `phase_correction`, described in the table below.

Required Input	Input type	Description
`fname`	String	Path and filename to the audio file to be analysed.
Optional parameter		
`dev_output`	Bool, default to `False`	When set to `False`, the code returns the predicted timbral value as a single float. When set to `True` returns the value of each extracted feature without the regression model applied.
`phase_correction`	Bool, default to `False`	When set to True, the phase of the left and right channels are compared before summing to mono (for stereo signals only). If the correlation is less than -0.5, the phase of the right channel is inverted before summing.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

The following subsections give more detail about the use of each timbral model, including other optional parameters.

1.2 Timbral Hardness

The timbral hardness model (Version 0.2) is a Python implementation model. This model has undergone improvements since the initial prototype described in Deliverable D5.2, extracting additional features to better fit with subjective ratings of perceived hardness.

1.2.1 General use of the Hardness model

The `timbral_hardness` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```python
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
hardness = timbral_models.timbral_hardness(fname)
```

This will return the estimated hardness of the audio file as a single floating point value. Higher values represent the audio file as sounding more *hard*. Although the model was trained on subjective data ranging from 0 to 100, the output values can be beyond these scales due to the nature of the linear regression implemented. The `clip_output` optional parameter can be used to limit the returned values from 0 to 100.

1.2.2 Additional features of the Hardness model

As well as the required filename, the timbral hardness model has the following variable parameters. Each parameter has been set to provide the highest correlation with subjective ratings.

Optional parameter	Parameter type	Description
`clip_output`	Bool, default to False.	Clips the output to between 0 and 100, the values used for training data.
`max_attack_time`	Float, default to 0.1	Sets the maximum period after each onset to evaluate the attack time, in seconds.
`bandwidth_thresh_db`	Float, default to -50	Sets the threshold for estimating the bandwidth of the audio signal, in dB.

1.3 Timbral Depth

The timbral depth model (Version 0.2) is a Python implementation model. Like with the Hardness model, this has undergone improvements since the initial prototype described in Deliverable D5.2, now extracting different signal features to better fit with subjective ratings of depth.

1.3.1 Using the Depth model

The `timbral_depth` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
depth = timbral_models.timbral_depth(fname)
```

This will return the estimated depth of the audio file as a single floating point value. Higher values represent the audio file as sounding more *deep*. Although the model was trained on subjective data ranging from 0 to 100, the output values can be beyond these scales due to the nature of the linear regression implemented. The `clip_output` optional parameter can be used to limit the returned values from 0 to 100.

1.3.2 Additional features of the Depth model

As well as the required filename, the timbral depth model has the following optional parameters. Each parameter has been set to provide the highest correlation with subjective ratings.

Optional parameter	Parameter type	Description
`clip_output`	Bool, default to False.	Clips the output to between 0 and 100, the values used for training data.
`threshold_db`	Float, default to -60	Threshold for estimating the spectral centroid features for a given spectrogram frame, in dB.
`low_frequency_limit`	Float, default to -50	Frequency to high-pass filter the audio signal, in Hz.
`centroid_crossover_frequency`	Float, default to 2000	Frequency of the crossover for calculating the spectral centroid features, in Hz.
`ratio_crossover_frequency`	Float, default to 500	Frequency of the crossover for calculating the spectral ratio features, in Hz.
`db_decay_threshold`	Float, default to -40	Threshold for estimating the decay time of the signal, in dB.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.4 Timbral Brightness

The timbral brightness model (Version 0.2) is a Python implementation model. Like with the hardness and depth models, this has undergone improvements since the initial prototype described in Deliverable D5.2, now extracting different signal features to better fit with subjective ratings of brightness.

1.4.1 Using the Brightness model

The `timbral_brightness` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
brightness = timbral_models.timbral_brightness(fname)
```

This will return the estimated brightness of the audio file as a single floating point value. Higher values represent the audio file as sounding more *bright*. Although the model was trained on subjective data ranging from 0 to 100, the output values can be beyond these scales due to the nature of the linear regression implemented. The `clip_output` optional parameter can be used to limit the returned values from 0 to 100.

1.4.2 Additional features of the Brightness model

As well as the required filename, the timbral brightness model has the following variable parameters. Each parameter has been set to provide the highest correlation with subjective ratings.

Optional parameter	Parameter type	Description
`clip_output`	Bool, default to False.	Clips the output to between 0 and 100, the values used for training data.
`threshold`	Float, default to 0	Threshold below which to ignore the energy in a time window.
`ratio_crossover`	Float, default to 2000	Crossover frequency for calculating the HF energy ratio in Hz.
`centroid_crossover`	Float, default to 100	Highpass frequency for calculating the spectral centroid in Hz.
`stepSize`	Int, default to 1024	Step size for calculating spectrogram, in samples.
`blockSize`	Int, default to 2048	Block size (fft size) for calculating spectrogram.
`minFreq`	Float, default to 20	Frequency for high-pass filtering audio prior to all analysis in Hz.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.5 Timbral Roughness

The timbral roughness model is in the process of being updated from the version released in D5.2. The version contained in this demonstrator (Version 0.2) is the same as that released in D5.2, modified to work with the new coding format.

1.5.1 Using the Roughness model

The `timbral_roughness` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
roughness = timbral_models.timbral_roughness(fname)
```

This will return the roughness estimated by the Vassilakis roughness formula [Vassilakis, 2007] as a single floating point value.

The `timbral_roughness` model does not have any optional parameters beyond those described in Section 1.1.2.

1.6 Timbral Warmth

The `timbral_warmth` model (Version 0.2) is a new addition to the `timbral_models` package. This model was created based on subjective ratings of warmth over 80 stimuli.

1.6.1 Using the Warmth model

The `timbral_warmth` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
warmth = timbral_models.timbral_warmth(fname)
```

This will return the estimated warmth of the audio file as a single floating point value. Higher values represent the audio file as sounding more '*warm*'. Although the model was trained on subjective data ranging from 0 to 100, the output values can be beyond these scales due to the nature of the linear regression implemented. The `clip_output` optional parameter can be used to limit the returned values from 0 to 100.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.6.2 Additional features of the Warmth model

As well as the required filename, the timbral warmth model has the following variable parameters. Each parameter has been set to provide the highest correlation with subjective ratings.

Parameter name	Parameter type	Description
`clip_output`	Bool, default to False.	Clips the output to between 0 and 100, the values used for training data.
`max_fft_frame_size`	Int, default to 8192	Frame size for calculating spectrogram, in Hz.
`rolloff_ratio`	Float, default to 0.85	Ratio for calculating the spectral rolloff frequency, defaults to 0.85 meaning 85% of the energy sits below this frequency.
`max_WR`	Float, default to 12000	Maximum allowable warmth region frequency (3.5 times the fundamental frequency), in Hz.

1.7 Timbral Sharpness

Another new model added to the `timbral_models` package is the `timbral_sharpness` model (Version 0.2). This is a direct implementation of the Fastl sharpness model [Fastl and Zwicker, 1991] and transcoded from the work of [Churchill, 2004]. This model still requires validation against subjective data.

1.7.1 Using the Sharpness model

The `timbral_sharpness` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
sharpness = timbral_models.timbral_sharpness(fname)
```

This will return the sharpness estimated by the Fastl sharpness algorithm [Fastl and Zwicker, 1991] as a single floating point value.

The `timbral_sharpness` model does not have any optional parameters beyond those described in Section 1.1.2.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

1.8 Timbral Booming

A third new model added to the `timbral_models` package is the `timbral_booming` model (Version 0.2). This is a direct implementation of the Hashimoto booming index [Hatano, 2000; Shin, 2009], designed to measuring the booming sensation inside moving cars.

1.8.1 Using the Booming model

The `timbral_booming` function can be called from the `timbral_models` package. The function should be given a string that is the path and filename of an audio file. For example:

```
import timbral_models
fname = '/Documents/Music/TestAudio.wav'
boominess = timbral_models.timbral_booming(fname)
```

This will return the boominess by the Hashimoto booming index algorithm [Hatano, 2000; Shin, 2009] as a single floating point value.

The `timbral_booming` model does not have any optional parameters beyond those described in Section 1.1.2.

1.9 Timbral Reverb

The feature extraction and modelling parts of our proposed approach are implemented in Matlab and Weka respectively. This is a newly developed approach to able us classify the audio files based on the perceived level of reverberation (ground truth came from the human subjective listening tests). Using this improved algorithm, the model better fits with subjective ratings of the perceived level of reverberation.

In this approach, new sets of features have been used. Like the Deliverable D5.2 report, one of extracted features is the RT60, but the extraction algorithm is different from what was proposed initially in D5.2. The new approach for extracting RT60 is based on blind estimation, meaning that it uses an audio signal as an input and not the room impulse response. This approach is based on the Laplacian model and the detail of this algorithm are given in [Jan and Wang, 2012].

In addition to RT60, the following features are also extracted and combined as an input for the machine learning modelling approaches:

- Level of foreground stream
- Level of background stream
- Interaural time difference (ITD) fluctuation in the foreground
- ITD fluctuation in the background
- Level of the low-frequency part of the spectrum
- Reverberance
- Clarity
- Apparent source width
- Listener Envelopment

These extra features are extracted as described in [Schuitman and Vries, 2013]. Schuitman et al. propose a method for derivation of signal based measures aiming at predicting aspects of room acoustic perception from content specific signal representations produced by a nonlinear model of the human auditory system. This approach simulates multiple aspects of human hearing to resemble human auditory perception.

1.9.1 Using the Reverb model

The `reverberation` functions which extract the features are called from the main Matlab script named `Reverb_AC_SS_V4`.

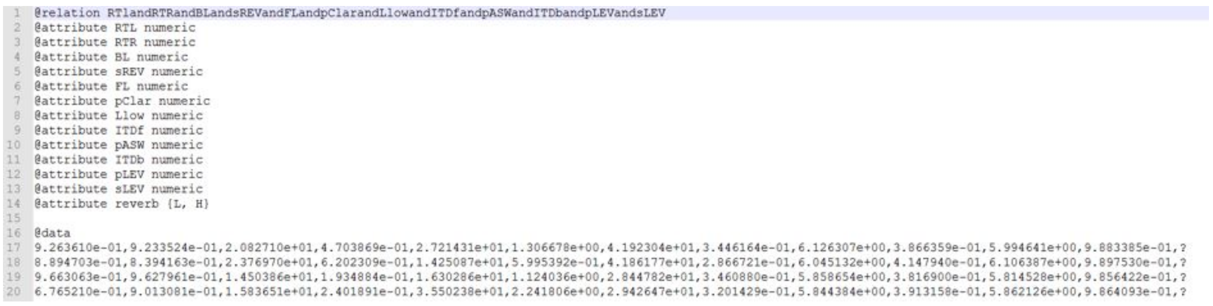
This file is in the project GitHub page and accessible from this location:
https://github.com/saeidsafavi/timbral_models/tree/patch-1/Reverb/V1

In the main script (`Reverb_AC_SS_V4.m`) couple of parameters needs to be defined: the path to the audio files; the name of the output file (the ".arff" file which will contains the extracted features); location of the ".arff" file which contains features of test audio files; location where the Weka is installed; and finally location and name of the trained reverberation model needs to be specified. For example:

```
Run Reverb_AC_SS_V1.m
Line 5: cd(' \\surrey.ac.uk\audiofiles\');
Line 8: fid = fopen('ExtractedFeatures.txt','w');
Line 39: cd('C:\Program Files\Weka-3-8');
Line 40: -l Logistic_classifier_Weka_D5.6.model -T
         C:\Users\Fred\Desktop\temp\outputfile.arff
```

By running the main script an explorer will be open in the location which contains all the audio files (the directory which is defined at line 5 of the main script) in the directory. The selected (multiple audio files could be selected at the same time) audio files then will be analysed and all the features will be extracted.

Extracted features will be stored in the comma separated format and in a single ".arff" file, with the name which is specified in the main Matlab script (line 8). Extracted features then need to be aligned and reformatted so that they can be used as an input for Weka. As it can be seen from the following figure, in the ".arff" files relation, attribute and data need to be defined. The following example shows that line 1 defines the relation between the features and lines 2 to 14 define the attributes. The extracted features for each audio files is stored from line 16 onwards. By running the main Matlab script the ".arff" file is generated for all the selected audio test files automatically.



```
1  @relation RTlandRTRandBLandsREVandFLandpClarandLlowandITDfandpASWandITDbandpLEVandsLEV
2  @attribute RTL numeric
3  @attribute RTR numeric
4  @attribute BL numeric
5  @attribute sREV numeric
6  @attribute FL numeric
7  @attribute pClar numeric
8  @attribute Llow numeric
9  @attribute ITDf numeric
10 @attribute pASW numeric
11 @attribute ITDb numeric
12 @attribute pLEV numeric
13 @attribute sLEV numeric
14 @attribute reverb {L, H}
15 
16 @data
17 9.263610e-01,9.233524e-01,2.082710e+01,4.703869e-01,2.721431e+01,1.306678e+00,4.192304e+01,3.446164e-01,6.126307e+00,3.866359e-01,5.994641e+00,9.883385e-01,?
18 8.894703e-01,8.394163e-01,2.376970e+01,6.202309e-01,1.425087e+01,5.995392e-01,4.186177e+01,2.866721e-01,6.045132e+00,4.147940e-01,6.106387e+00,9.897530e-01,?
19 9.663063e-01,9.627961e-01,1.450386e+01,1.934884e-01,1.630286e+01,1.124036e+00,2.844782e+01,3.460880e-01,5.858654e+00,3.816900e-01,5.814528e+00,9.856422e-01,?
20 6.765210e-01,9.013081e-01,1.583651e+01,2.401891e-01,3.550238e+01,2.241806e+00,2.942647e+01,3.201429e-01,5.844384e+00,3.913158e-01,5.862126e+00,9.864093e-01,?
```

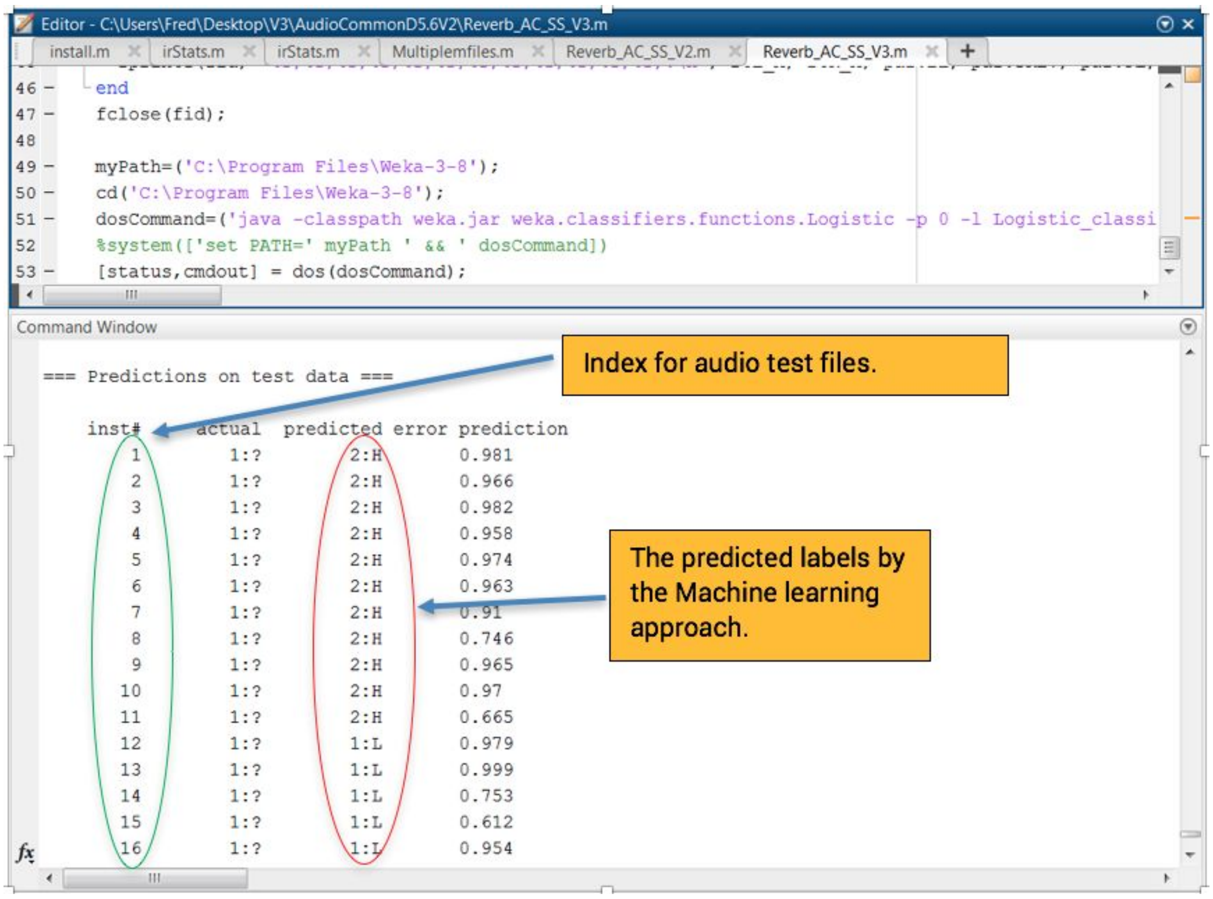
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

To identify the prediction output, two class labels are defined during the training phase. During the training the audio files are labelled with L or H. These labels are came from the human listening tests, in which L means the audio file has low level of reverberation and H means High level.

The Reverberation model is trained using 420 audio files from multiple source types, and simple logistic classifier (Weka logistic classifier with the default predefined parameters) is used for modelling. Details about this classifier could be found in [Le Cessie and van Houwelingen, 1992].

The pre-trained reverberation model and generated ".arff" files can now be used in the Weka interface for evaluation. Weka could be used via its own graphic interface or from the Command Prompt environment. In this report, to make it simple for evaluation, the Weka evaluation part is also integrated within the Matlab main script (`Reverb_AC_SS_V4.m`). By running the main script, the output will be the prediction for each of the audio files.

The output of the Matlab script has a format like this:


Editor - C:\Users\Fred\Desktop\V3\AudioCommonD5.6V2\Reverb_AC_SS_V3.m
install.m
irStats.m
irStats.m
Multiplemfiles.m
Reverb_AC_SS_V2.m
Reverb_AC_SS_V3.m
Command Window
Index for audio test files.
The predicted labels by the Machine learning approach.

```
    end
    fclose(fid);

    myPath=('C:\Program Files\Weka-3-8');
    cd('C:\Program Files\Weka-3-8');
    dosCommand=('java -classpath weka.jar weka.classifiers.functions.Logistic -p 0 -l Logistic_classi
    %system(['set PATH=' myPath ' && ' dosCommand])
    [status,cmdout] = dos(dosCommand);
```

```
=== Predictions on test data ===

    inst#     actual  predicted error prediction
        1        1:?        2:H       0.981
        2        1:?        2:H       0.966
        3        1:?        2:H       0.982
        4        1:?        2:H       0.958
        5        1:?        2:H       0.974
        6        1:?        2:H       0.963
        7        1:?        2:H       0.91
        8        1:?        2:H       0.746
        9        1:?        2:H       0.965
       10        1:?        2:H       0.97
       11        1:?        2:H       0.665
       12        1:?        1:L       0.979
       13        1:?        1:L       0.999
       14        1:?        1:L       0.753
       15        1:?        1:L       0.612
       16        1:?        1:L       0.954
```

In addition to test index and predicted labels there also is a column which shows the probability of this prediction based on the predefined model. Full description of this approach and the obtained results described fully in [Safavi, Pearce, Wang, and Plumbley, 2018].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

2 Conclusion

In this deliverable, the implementations of timbral models of *hardness*, *depth*, *brightness*, *roughness*, *warmth*, *sharpness*, *boominess*, and *reverb* were discussed. These updated and new models will be incorporated into the AudioCommons ecosystem as a means of automatically generating metadata that can be used to supplement searches, making it easier for users to identify suitable sound effects.

The models of *hardness*, *depth*, and *brightness* have all been remodelled to better reflect subjective ratings of the attributes. The model of roughness is currently being evaluated against newly collected subjective data and the evaluation will be submitted as part of Deliverable D5.7. The three new models of *warmth*, *sharpness*, and *boominess* will also be evaluated against a larger set of subjective ratings in D5.7, with the models being improved and refined as necessary.

Following this, final implementations of these timbral models will be developed, improving their computational efficiency and accuracy to their subjective attributes, for release in Deliverable D5.8. This release will also import the reverb model into a python executable format, and include a `timbral_extractor` function that will allow the predictions of each timbral attribute to be calculated with a single function call.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 688382

3 References

Churchill, C., 2004: “MATLAB Codes, Calculating the Metrics”, Salford Innovation Research Centre, https://www.salford.ac.uk/research/sirc/research-groups/acoustics/psychoacoustics/sound-quality-making-products-sound-better/accordion/sound-quality-testing/matlab-codes.

Fastl, E., and Zwicker, H.,1991: “Psychoacoustics, Facts and Models”, Springer.

Hatano, S., and Hashimoto, T., 2000: "Booming index as a measure for evaluating booming sensation", Internoise 2000: 29th International congress and Exhibition on Noise Control Engineering, pp. 4332-4336.

Jan, T., and Wang, W., 2012: “Blind reverberation time estimation based on Laplace distribution”, *EUSIPCO*. pp. 2050-2054, Bucharest, Romania.

Le Cessie, S. and van Houwelingen, J., 1992: “Ridge estimators in logistic regression”, Applied Statistics, Vol. 41, No. 1, pp. 191-201.

Safavi, S., Pearce, A., Wang, W., & Plumbley, M. (2018). Predicting the perceived level of reverberation using machine learning. (Accepted for) *Asilomar conference on signals, systems and computers*. Pacific Grove.

Schuitman, J., and Vries, D., 2013: “Deriving content-specific measures of room acoustic perception using a binaural, non linear auditory model”. *Acoustical society of America*, 133(3), pp.1572-1585.

Shin, S-H., Ih, J-G., Hashimoto, T., and Hatano, S., 2009: "Sound quality evaluation of the booming sensation for passenger cars", Applied Acoustics, Vol. 70(2), pp 309-320.

Vassilakis, P., 2007: “SRA: A web-based research tool for spectral and roughness analysis of sound signals”, in Proceedings of 4th Sound Music Computing (SMC), pp. 319–325.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement N° 688382