AMUSE (ADVANCED MUSIC EXPLORER) - A MULTITOOL FRAMEWORK FOR MUSIC DATA ANALYSIS

Igor Vatolkin
Chair of Algorithm Engineering,
TU Dortmund
igor.vatolkin@udo.edu

Wolfgang Theimer
Research in Motion, Bochum
wolfgang.theimer@ieee.org

Martin Botteck
martin.botteck@ieee.org

ABSTRACT

A large variety of research tools is available now for music information retrieval tasks. In this paper we present a further framework which aims to facilitate the interaction between these applications. Since the available tools are very different in target domain, range of available methods, learning efforts, installation and runtime characteristics etc., it is not easy to find software which is optimal for certain research goals. Another problematic issue is that many incompatible data formats exist, so it is not always possible to use output from one tool just as input for another one. At first we describe some of the available projects and outline our motivation starting the development of AMUSE framework for audio data analysis. Requirements and application purposes are given. The structure of our framework is introduced in detail and the information for efficient application is provided. Finally we discuss several ideas for further work.

1. INTRODUCTION AND MOTIVATION FOR A NEW FRAMEWORK

During the recent years more and more scientific tools for music information retrieval and related research areas have been developed. To name just a few, Marsyas is one of the oldest available MIR projects for different analysis and synthesis tasks [12]. jMIR tools refer to different applications from feature extraction to data mining methods [7]. MusicMiner established new navigation techniques for large music collections based on self-organized maps and three-dimensional landscapes [9]. MIR Toolbox includes a large number of different adjustable Matlab functions for extraction of features from time signal characteristics to complex harmony and major/minor key descriptors [4]. The Chroma Toolbox provides advanced features related to chroma and pitch [10]. RapidMiner is aimed to solve a wide range of different data mining tasks (not only for music and audio classification domain) and supports numerous methods [8].

The motivation to start our own project developing a new software framework arose after the discussion and definition of several promising MIR applications and in-depth comparison of different above mentioned and further tools. Typically each existing tool has several main focus points as well as certain application advantages and disadvantages. Therefore the choice of software to use depends strongly on the defined scenario. Possible three examples could be: If the researcher develops own classification methods, it may be interesting for her/him to gather many available audio features from the corresponding tools. If the aim is to run advanced low-level signal analysis and create the features by himself, the researcher would create this code and use some ready products, e.g. WEKA toolbox [13] for the revision, how well the novel features are suited for audio classification. The last example is that for different reasons multi-objective evaluation of algorithmic chain can be significant. Here the focus point is to collect different metrics (confusion matrix-based measures, runtime and disc space demands etc.) and to run multi-objective optimization algorithms searching for the best tradeoff between several solutions.

Another aspect is that many tools which are very helpful for MIR research are either too specific and concentrate on limited audio retrieval domains, e.g. Chroma Toolbox (so we may need several of them!) or are on the other side too powerful and generic (e.g. RapidMiner) and it is not easy to create the appropriate solution. The input and output data formats differ from tool to tool and even the support of the WEKA ARFF format does not mean, that the written attributes are the same. Therefore it made sense for us to develop a multi-tool framework, which provides own data interchange formats and own evaluation methods. The integration of further tools and also the extension of methods with own code belonged to requirements. Further consideration was that some fields had been underrepresented in many available tools and it was important for us to emphasize them as independent tasks in music retrieval chain: feature processing is an intermediate step between the extracted raw features and ready classifier input. The way how the labeled vector is built from the frame-based signal features for training of classification models can be very different and has a strong impact on classification result quality. Another issue is the inclusion of optimization methods, e.g. heuristics, to search for the best parametrization of the algorithm chain, for example the estimation of satisfactory time frame size or pruning of feature set.

The current version of AMUSE made possible to run different large experiment studies including feature extraction from several tools, processing with many methods, classification for user-defined music categories and also optimization of some parameters [3, 14, 15].

2. FRAMEWORK STRUCTURE

2.1 Background, Requirements and Functionality

AMUSE (Advanced MUSic Explorer) is a GPL-licensed framework implemented in Java [1] . Therefore the main component can be run on any operating system which supports the Java Runtime Environment. The integrated tools have no usage restrictions with regard to their source codes. If they are not available as Java libraries, executable versions must be provided. In that case it may certainly lead to the dependence on the running operating system.

The AMUSE core provides different functionalities. With own sound processing methods mp3s can be converted to waves. Downsampling and stereo to mono conversion methods are included as well. It is possible to split automaticly the wave files (we had experiences that very long songs supplied to some tools led to memory problems or unacceptable running time). Scalability is supported either using multi-threading on one machine or providing the tasks to grid systems like Sun Grid Engine or LSF Batch. Efficient data set management which directly supports the WEKA ARFF format (well-established for various data mining tasks) and a logger component are integrated.

Several user interfaces are available: Definition and application of tasks can be easily done within a graphical user interface, see Figure 5 for screenshots. In command-line mode AMUSE runs one or more tasks from given configuration files. In loop mode AMUSE is pre-loaded in memory and waits for new tasks by scanning for corresponding configuration files in a task folder.

Project packages are organized in the way so that the core and extendable components are strictly separated. Integration of external tools requires writing of adapter classes which take care of input / output conversion and start these tools as library or by system call. AMUSE plugins allow to create such integrations without changes on the main project and be easily installed and deinstalled.

2.2 Music Retrieval Chain and Integrated Methods

We distinguish between subtasks in a MIR chain. Figure 1 gives a complete overview. The rectangles correspond to AMUSE *tasks* which are run by the related node component. Each task can adhere to the larger number of AMUSE *jobs* which can be calculated on several processing units - e.g. a feature extraction task for a hundred music files can be distributed to several machines as one hundred jobs.

2.2.1 Feature Extraction

Feature extraction provides low-level or high-level numerical descriptors from the audio signal. It can be a complete task (melody extraction) or a part of a longer chain, where audio files are categorized using the extracted features. AMUSE provides a generic mechanism to select the features which must be extracted by external tools. For each tool a so called *base script* must exist which allows to extract all supported features. After the AMUSE extraction task is loaded into memory, some parts of these base scripts are omitted, if the corresponding transforms or features should not be extracted this time.

2.2.2 Feature Processing

Feature processing is an intermediate step between raw extracted features and ready-labeled input for classification. Starting with a matrix of M features over N time frames at the beginning, different methods change this matrix. Some of them extend the dimensionality (e.g. calculating the derivations for all features) or reduce the dimensionality (pruning the features or deselecting time frames using specific information like temporal structure of a song). The last step is the conversion of the feature matrix to a vector which can be labeled for supervised classification. This can be done e.g. by Gaussian, histogram or autoregressive models. Since the source time frames may differ between features, the matrix is automaticly adjusted using the smallest existing time frame. For example if the first feature is calculated from 1024 sample frames and the second one from larger windows (number of beats per minute) and the third from the complete song (music track length), N will be set to the number of 1024 sample frames in the complete song. A music track length feature will then have the same values for all corresponding matrix entries.

2.2.3 Classification and Training

Supervised classification training creates models from given data and requires the ground truth information. Classification applies the previously learned models and computes the list of relations to the given categories for the provided music tracks. Unsupervised classification techniques categorize data without any given information by e.g. clustering. It is possible to run preprocessing before the classification, for example removing the outliers. Since RapidMiner [8] and WEKA [13] are also Java-based projects, they are integrated into AMUSE directly as libraries. It is also possible to connect to Matlab or R engines starting further classification methods.

2.2.4 Validation

The classification validator is responsible for evaluation of classification results. Confusion matrix-based metrics and error rates are well known and measure the quality of classification results. Other measures relate to the balance aspect - if a data set contains too much positive instances, accuracy may be high inspite of the poorly designed algorithm which tends to categorize everything as positive. Further it is possible to measure correlation between ground truth and predicted category relationships. All these metrics can be either calculated on the lower data level (measuring the classification success for smaller audio intervals as data instances) or on the higher data level

[1] http://amuse-framework.sourceforge.net

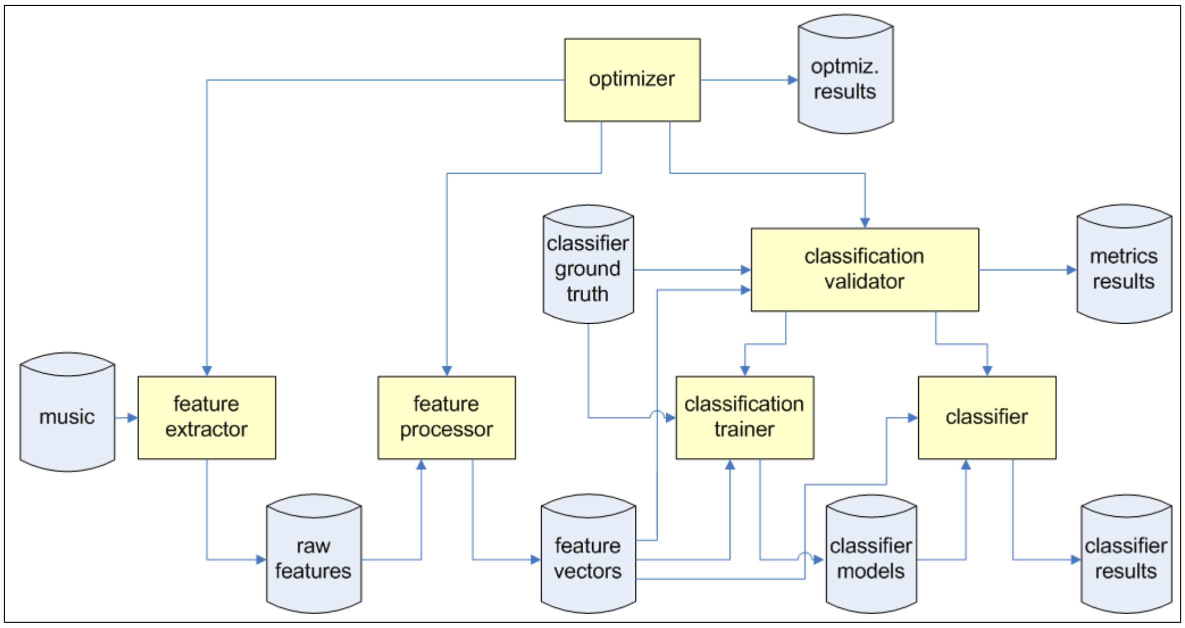


Figure 1. AMUSE task chain.

(evaluating classification averaged for the complete music tracks). Other metric group (data reduction rate) calculates the amount of data used for training related to the size of the original feature matrix.

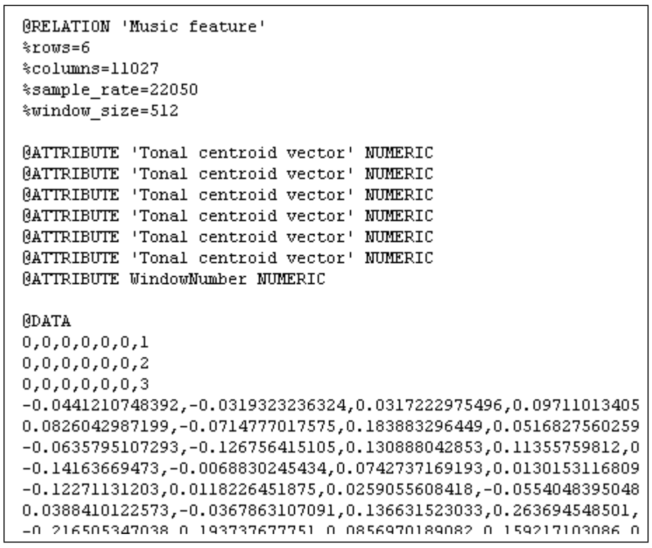
2.2.5 *Optimization*

Each of the above mentioned tasks has more or less parameters and it is obviously, that it is not a simple task to find an optimal combination of them. For example the larger time frames for low-level feature extraction allow a more precise frequency resolution up to the Nyquist frequency. But if they are too long, different notes are mixed together and it becomes harder to learn anything. The optimization of the music data analysis chain is very rarely supported by related MIR tools. Indeed many optimization toolboxes exist (e.g. CILIB [11], SPOT [1] etc.) but the application is often too generic and must be adapted to the MIR domain. Therefore the goal of the optimization node is to run methods searching for optimal parameter settings. Currently several evolution strategies [2] are directly implemented in AMUSE and can be used for optimization.

2.3 Database Structure and Data Formats

Information provided by user (ground truth, music tracks) and the generated output are stored in folders called AMUSE databases. Most of the data is currently saved as text ARFF format [13] for several reasons: It is very comprehensible, is supported by most tools and is much more compact than XML. However it is an option to support further formats in future using e.g. MySQL database which requires more storage place but provides a very fast searching routine.

Music and *category* folders store the songs provided by the user and the corresponding ground truth for any related categories (music genres, information about harmony and melody etc.). *Feature* folders save the extracted features, the folder *processed features* stores the unlabeled feature vectors for classification. Binary classification models are placed in the *model* directory, *metric* database is used for evaluation of music data analysis experiments. *Optimization* database contains of optimization logs in search for optimal parameter settings. Currently AMUSE does not support any visualization methods, but the data can be easily read into well-established tools like Gnuplot or Matlab.



```
@RELATION 'Music feature'
%rows=6
%columns=11027
%sample_rate=22050
%window_size=512

@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE 'Tonal centroid vector' NUMERIC
@ATTRIBUTE WindowNumber NUMERIC

@DATA
0,0,0,0,0,0,1
0,0,0,0,0,0,2
0,0,0,0,0,0,3
-0.0441210748392,-0.0319323236324,0.0317222975496,0.09711013405
0.0826042987199,-0.0714777017575,0.183883296449,0.0516827560259
-0.0635795107293,-0.126756415105,0.130888042853,0.11355759812,0
-0.14163669473,-0.0068830245434,0.0742737169193,0.0130153116809
-0.12271131203,0.0118226451875,0.0259055608418,-0.0554048395048
0.0388410122573,-0.0367863107091,0.13663152303 3,0.26369454850 1,
-0.21650534703 8,0.19373767775 1,0.08569701890 82,0.15921710308 6,0
```

Figure 2. Example ARFF file with extracted features.

An example for a feature file is given in Figure 2. Here the extended ARFF format is used: AMUSE attributes are placed as comments after the relation description and store the information about the data set size, sampling frequency and time frame size. Since for each feature the corresponding time frame is saved in attribute WindowNumber, it is simple to detect the time intervals from which the features have been extracted.

The AMUSE experiments are also saved as ARFFs.

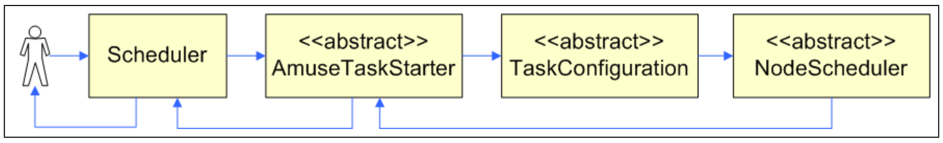

Scheduler
<<abstract>> AmuseTaskStarter
<<abstract>> TaskConfiguration
<<abstract>> NodeScheduler

Figure 3. Data flow in AMUSE.

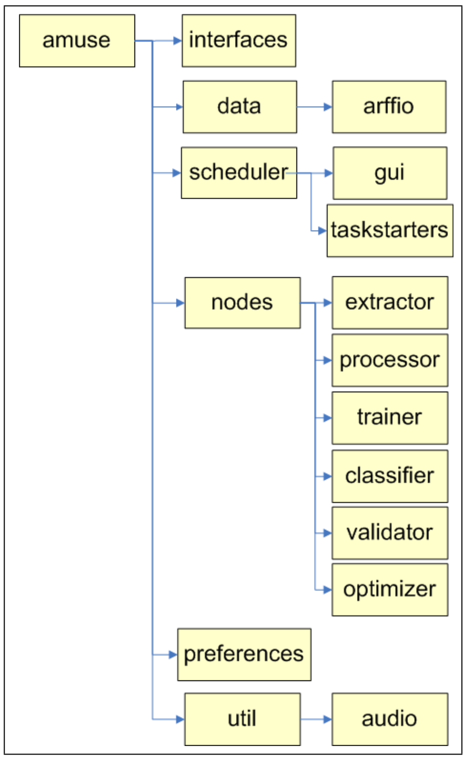


Figure 4. Package structure.

3. DETAILS FOR DEVELOPERS

The data flow during an AMUSE experiment is shortly depicted in Figure 3. User starts the main SCHEDULER component either from GUI or from the command line. After the configuration of the experiment the tasks are completely described in the corresponding TASKCONFIGURATION objects which are provided to the appropriate TASKSTARTER component. Here one or more AMUSE jobs are generated. These jobs are run on the same machine or are processed to a grid system. During the runtime of a single AMUSE instance the scheduler counts up the jobs. After they are ready, the next experiment can be started.

3.1 Package Structure

The most important AMUSE Java packages are shown in Figure 4. Scheduler and GUI packages interact with user, the computing of jobs is done in objects which are placed in nodes package and extend an abstract class NODESCHEDULER. Data package handles AMUSE data objects and ARFF input / output routines. Preferences store different configuration parameters (database folders, downsampling rate, path to grid scripts etc.). The util package contains logger and audio processing methods.

3.2 Guidelines for Tool Integration

Here we give a brief overview for several steps needed to extend AMUSE with new tools:

- Tool setup: Software which should be integrated into AMUSE must be tested for execution on the current operating system. It must be possible to start it either as Java library or by system call using a previously configured batch file.

- Writing an adapter class: Here the tool will be started. The functions which convert input and output data to AMUSE format must be implemented. If e.g. a feature extractor program saves the data as XML, it has to be converted into ARFF format for features as given in Figure 2.

- Plugin definition: The default way to integrate a new tool into AMUSE is to create the corresponding plugin. Several plugin installation files (mostly ARFFs) must describe the changes in AMUSE which will be applied after the installation. Each feature and each method used in AMUSE has a unique id number. The configuration file featureTable.arff lists all currently available features with given ids. If a new tool allows the extraction of several new features, this file must be updated. The same procedure is essential for further algorithms. There is a list with all available classification methods, validation metrics, processing and preprocessing algorithms etc.

- Plugin installation and integration should be tested. After the successful evaluation the job is done!

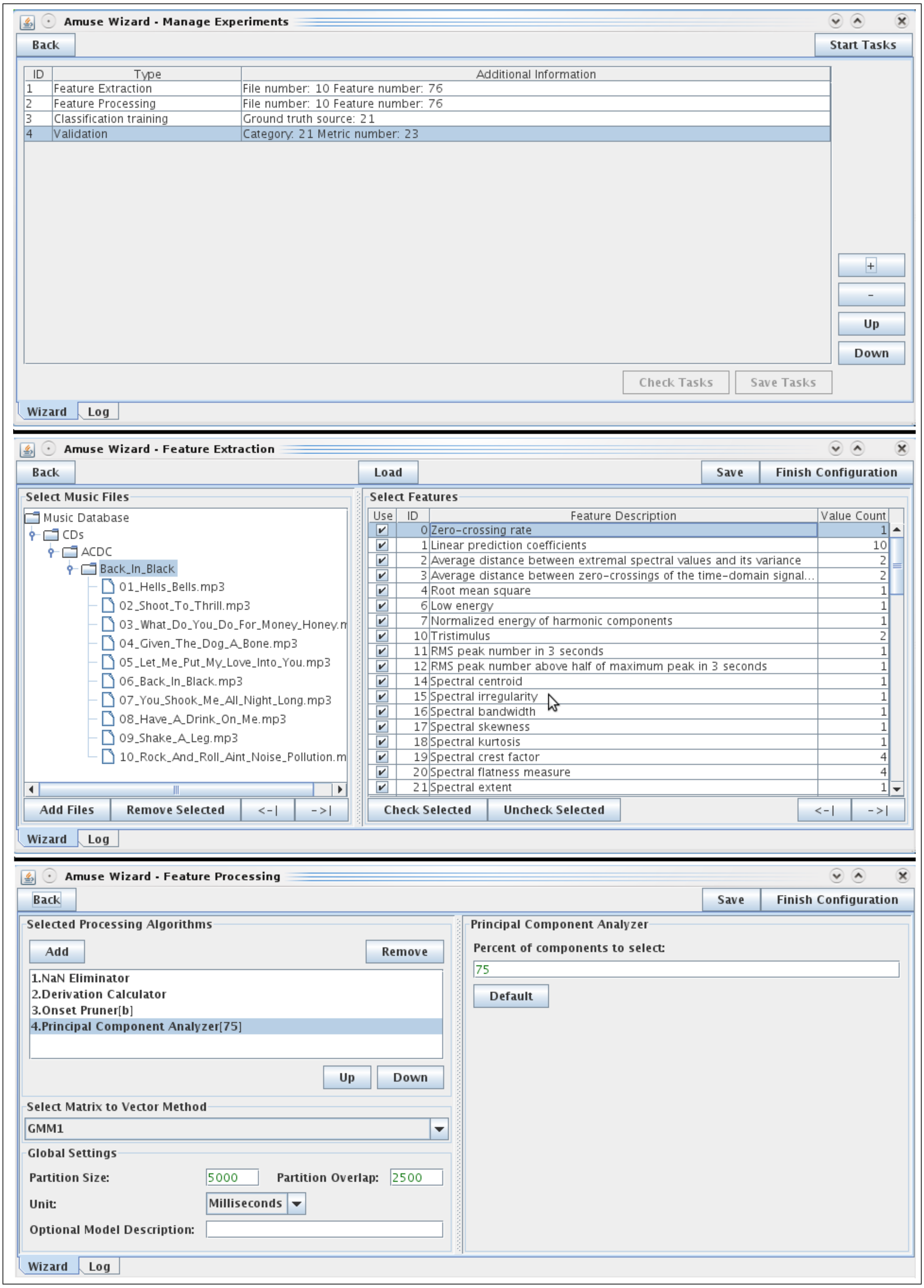
4. ONGOING WORK

The core framework has been already developed, however a lot of work remains. In the near future we will provide comprehensive introduction and developer manuals. Integration of further tools and extension with own methods belongs to the current and ongoing activities. Especially the optimization node will be extended with new methods related to multi-objective evaluation and computational intelligence algorithms.

As further steps we plan to add some visualization possibilities for experiment results and navigation possibilites through given music collections. The algorithms for symbolic and community-based retrieval can be also integrated.

5. ACKNOWLEDGEMENTS

We thank the Klaus Tschira Foundation for the financial support.



Amuse Wizard - Manage Experiments

Back | Start Tasks

ID	Type	Additional Information
1	Feature Extraction	File number: 10 Feature number: 76
2	Feature Processing	File number: 10 Feature number: 76
3	Classification training	Ground truth source: 21
4	Validation	Category: 21 Metric number: 23

Amuse Wizard - Feature Extraction

Back | Load | Save | Finish Configuration

Select Music Files

```
Music Database
  CDs
    ACDC
      Back_In_Black
        01_Hells_Bells.mp3
        02_Shoot_To_Thrill.mp3
        03_What_Do_You_Do_For_Money_Honey.m
        04_Given_The_Dog_A_Bone.mp3
        05_Let_Me_Put_My_Love_Into_You.mp3
        06_Back_In_Black.mp3
        07_You_Shook_Me_All_Night_Long.mp3
        08_Have_A_Drink_On_Me.mp3
        09_Shake_A_Leg.mp3
        10_Rock_And_Roll_Aint_Noise_Pollution.m
```

Select Features

Use	ID	Feature Description	Value Count
✓	0	Zero-crossing rate	1
✓	1	Linear prediction coefficients	10
✓	2	Average distance between extremal spectral values and its variance	2
✓	3	Average distance between zero-crossings of the time-domain signal...	2
✓	4	Root mean square	1
✓	6	Low energy	1
✓	7	Normalized energy of harmonic components	1
✓	10	Tristimulus	2
✓	11	RMS peak number in 3 seconds	1
✓	12	RMS peak number above half of maximum peak in 3 seconds	1
✓	14	Spectral centroid	1
✓	15	Spectral irregularity	1
✓	16	Spectral bandwidth	1
✓	17	Spectral skewness	1
✓	18	Spectral kurtosis	1
✓	19	Spectral crest factor	4
✓	20	Spectral flatness measure	4
✓	21	Spectral extent	1

Amuse Wizard - Feature Processing

Back | Save | Finish Configuration

Selected Processing Algorithms

1.NaN Eliminator
2.Derivation Calculator
3.Onset Pruner[b]
4.Principal Component Analyzer[75]

Select Matrix to Vector Method: GMM1

Global Settings

Partition Size: 5000 | Partition Overlap: 2500
Unit: Milliseconds
Optional Model Description:

Principal Component Analyzer

Percent of components to select: 75

Figure 5. AMUSE GUI: Management of experiments (top); Feature extraction experiment setup (middle); Feature processing experiment setup (bottom).

6. APPENDIX: LIST WITH INTEGRATED TOOLS

Here we give an alphabetically sorted list of currently integrated tools. In AMUSE it is easy to create complex experiments using different algorithms, e.g. extracting features with jAudio and MIR Toolbox, preprocessing them with Matlab, classifying with WEKA and validating them with metrics available in AMUSE.

- Chroma Toolbox: Extraction of different novel chroma and pitch features [10].
- CMRARE: A set of cepstral modulation ratio regression (CMRARE) parameters for audio signal [5].
- jAudio: Java application for audio feature extraction [6].
- Matlab: Corresponding AMUSE adapter allows to run Matlab code. Path to the installed Matlab version must be set in AMUSE configuration.
- MIR Toolbox: A large set of Matlab fuctions for extraction of low-level and high-level audio descriptors [4].
- R: Corresponding AMUSE adapter allows to run R code. Path to the installed R version must be set in AMUSE configuration.
- RapidMiner (former Yale): Java framework for data mining [8]. A large number of different classification and data processing algorithms is available, audio feature extraction is provided by ValueSeries plugin.
- WEKA: An established framework for machine learning which is integrated as library in RapidMiner [13].

7. REFERENCES

[1] T. Bartz-Beielstein: *Experimental Research in Evolutionary Computation - The New Experimentalism*, Springer Verlag, 2006.

[2] H.-G. Beyer and H.-P. Schwefel: "Evolution Strategies - A Comprehensive Introduction," *Natural Computing*, Vol. 1, No. 1, pp. 3–52, 2002.

[3] B. Bischl, I. Vatolkin and M. Preuss: "Selecting Small Audio Feature Sets in Music Classification by Means of Asymmetric Mutation," Accepted for the *11th International Conference on Parallel Problem Solving from Nature (PPSN)*, Krakow, 2010.

[4] O. Lartillot and P. Toiviainen: "MIR in Matlab (II): A Toolbox for Musical Feature Extraction From Audio," *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)* pp. 127–130, 2007.

[5] R. Martin and A. Nagathil: "Cepstral Modulation Ratio Regression (CMRARE) Parameters for Audio Signal Analysis and Classification," *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 321–324, 2009.

[6] D. McEnnis, C. McKay and I. Fujinaga: "jAudio: Additions and Improvements," *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR)*, pp. 385–386, 2006.

[7] C. McKay and I. Fujinaga: "jMIR: Tools for Automatic Music Classification," *Proceedings of the International Computer Music Conference (ICMC)*, pp. 65-68, 2009.

[8] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler: "YALE: Rapid Prototyping for Complex Data Mining Tasks," *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06)*, pp. 935–940,2006.

[9] F. Mörchen, A. Ultsch, M. Noecker, C. Stamm: "Databionic Visualization of Music Collections According to Perceptual Distance," *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, pp. 396–403, 2005.

[10] M. Müller: *Information Retrieval for Music and Motion*, Springer Verlag, 2007.

[11] G. Pamparà, A.P. Engelbrecht and T. Cloete: "CIlib: A collaborative framework for Computational Intelligence algorithms - Part I," *Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI)*, pp. 1750–1757, 2008.

[12] G. Tzanetakis and P. Cook: "Marsyas: A framework for Audio Analysis," *Organised Sound*, Vol. 4, No. 3, pp. 169–175, 2000.

[13] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes and S.J. Cunningham: "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," *Proceedings of the ICONIP/ANZIIS/ANNES'99 Workshop on Emerging Knowledge Engineering and Connectionist-Based Information Systems*, pp. 192–196, 1999.

[14] I. Vatolkin and W. Theimer: "Optimization of Feature Processing Chain in Music Classification by Evolution Strategies," *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN)*, Dortmund, pp. 1150-1159, 2008.

[15] I. Vatolkin, W. Theimer and G. Rudolph: "Design and Comparison of Different Evolution Strategies for Feature Selection and Consolidation in Music Classification," *Proceedings of the 2009 IEEE Congress on Evolutionary Computation (CEC 2009)*, IEEE Press, Piscataway (NJ), 2009.