JMIR: TOOLS FOR AUTOMATIC MUSIC CLASSIFICATION

Cory McKay and Ichiro Fujinaga

McGill University
Music Technology Area
cory.mckay@mail.mcgill.ca, ich@music.mcgill.ca

ABSTRACT

jMIR is a free and open-source software suite designed for applications related to automatic music classification. jMIR includes the jAudio, jSymbolic and jWebMiner feature extractors, the ACE meta-learning framework, the ACE XML information exchange file formats, the jMusicMetaManager musical dataset management software and the Codaich, Bodhidharma MIDI and SAC musical datasets. The primary goals underlying jMIR are the provision of powerful ready-to-use software tools to music researchers with diverse ranges of technical backgrounds, the encouragement of research combining features derived from audio, symbolic and cultural sources of data, and the provision of a framework for iteratively and collaboratively developing further music information retrieval (MIR) tools and performing original music classification research.

1. INTRODUCTION

Many MIR research areas are strongly dependent upon the three central tasks of collecting and annotating ground truth data, extracting characteristic features from this data, and applying pattern recognition algorithms to the results. Such research areas include automatic genre classification, mood classification, performer or composer identification, music recommendation, hit prediction, and instrument identification, to name only a few. Furthermore, many of the most significant developments in MIR have involved innovation in one or more of these key areas.

Unfortunately, the lack of general, flexible, standardized and easy-to-use tools designed specifically for MIR research has made it difficult to compare, evaluate, share and build upon the approaches used by different researchers. This has necessitated a significant amount of duplicated effort and a resultant slowdown in the progress of MIR research.

MIR researchers need software tools for quickly and easily performing the basic tasks underlying MIR research, and they need frameworks for developing new techniques that can in turn be used to iteratively develop additional techniques. It should be a simple matter, for example, to use existing software to extract common features such as MFCCs and then, in the same software framework, use them to design new higher-level features, such as pitch-related features. There should also be an infrastructure for distributing data, annotations and implementations of new algorithms to other researchers in a standardized way so that they can quickly evaluate them and integrate them into their own research as software plug-ins.

Contrary to this ideal, it is much more common to find each MIR research group working with its own in-house software and musical datasets. Efforts to make the means and fruit of each group's research easily usable by others are unfortunately often only an afterthought. For example, the researchers running the annual MIREX MIR competition (www.music-ir.org/mirex/2009/) have found that submitted systems rarely work in their originally submitted form, even when required by the evaluators to meet set basic interface requirements.

jMIR has been developed in order to address these issues. jMIR is an open-source Java software suite consisting of tools for performing the primary tasks associated with automatic music classification research. It can be used to study music in both audio and symbolic forms, as well as to mine cultural information from the Internet. It includes software for extracting features, applying machine learning algorithms, and analyzing and managing metadata associated with musical datasets. The ACE XML file formats are also part of jMIR, and are intended not only for use with the jMIR software, but also as an expressive and flexible general standard for exchanging and storing MIR-related information.

There are three primary priorities underpinning the design of the jMIR components. The first is that they be easy to install and use by individuals with a variety of technical backgrounds. This is essential, as researchers in fields such as music theory, psychology, library sciences and musicology have important musical insights that can greatly benefit MIR research, but are often alienated by software requiring a strong technical background to use. Installation difficulties and steep learning curves can also be discouraging for even technically skilled users. The jMIR components are consequently well-documented, relatively simple to install, and include easy-to-use GUIs.

The second priority is the provision of an open framework for performing novel research and distributing new approaches to others. This is important in ensuring

research transparency and in allowing researchers to evaluate and build upon each other's work. In order to accomplish this, the jMIR components are designed using a modular plug-in approach. It is thus a simple matter for researchers to both develop their own algorithms and add algorithms newly implemented by others to their jMIR distributions. The jMIR feature extractors also automatically make the value of each extracted feature available to each other feature and automatically consider dependencies when dynamically scheduling extraction order, something that facilitates iterative feature design.

The third priority is the provision of functionality allowing features extracted from audio recordings, symbolic representations and cultural data available on the Internet to be combined. Music researchers traditionally tend to focus on only one of these domains, and thus risk missing out on valuable complementary sources of information. It has been found experimentally using jMIR that combining these sources of information can improve music classification performance significantly [10].

As the details of earlier versions of most of the jMIR components have been individually published previously, primarily at the ISMIR conference, this publication focuses on introducing new developments and on presenting jMIR to the wider computer music community. This paper also highlights jMIR as a unified package that is more than just a collection of isolated tools. Although the jMIR components can certainly each be used separately, using them together can be advantageous.

The jMIR software and related documentation may be downloaded from jmir.sourceforge.net.

2. RELATED WORK

To the best of the authours' knowledge, jMIR is the only unified MIR system that encompasses symbolic, audio and cultural feature extraction, meta-learning, metadata management and standardized file formats.

There are, however, several high-quality toolsets that have been designed for broad MIR use. MIDIToolbox [2] and MIRToolbox [3] are powerful modular Matlab toolboxes for designing and extracting symbolic and audio features, respectively. The well-known CLAM [1] and Marsyas [12] systems focus on audio-related tasks. Also, the M2K (www.music-ir.org/evaluation/m2k/) graphical patching interface can be used to connect a range of different MIR processing components in ways that can take advantage of distributed processing.

3. THE JMIR COMPONENTS

3.1. jAudio

jAudio [4] is an application for extracting features from audio files in formats such as MP3, AIFF and WAV. It is bundled with 28 implemented features extracted from both the frequency and time domains (e.g., Spectral Flux, RMS, etc.). It includes several relatively high-level musical features, mainly related to rhythm, as well as lower-level signal processing-oriented features. A variety of pre-processing options are also available, such as downsampling, normalization and windowing.

In order to make jAudio as accessible as possible, it includes a GUI interface, a command-line interface, a Java API to facilitate integration with other software and batch file processing functionality. jAudio also includes functionality for synthesizing, recording and displaying audio in order to facilitate the testing of new features.

jAudio development since its original publication has focused on facilitating the process of developing and adding new features. As is also the case with jSymbolic, new features can be added to jAudio using a simple inheritance-based plug-in approach that automatically and dynamically solves scheduling dependencies. jAudio also includes implementations of "metafeatures" and "aggregators" that respectively automatically implement features derived from other features (e.g., the standard deviation of a feature across analysis windows) and collapse a set of feature vectors into a single vector or a smaller set of vectors (e.g., area of moments).

3.2. jSymbolic

jSymbolic [7] is a GUI-based application for extracting features from MIDI files. It is bundled with 111 implemented features (e.g., Note Density, Instruments Present, Range, etc.), most of which are based on relatively high-level musical abstractions and many of which are novel. This is far more features than any other symbolic feature extractor. The features fall into the broad categories of instrumentation, texture, rhythm, dynamics, pitch statistics and melody.

Like jAudio, jSymbolic has a simple inheritance-based modular API for adding new features, and feature dependencies are resolved automatically in order to encourage the iterative development of increasingly high-level features (e.g., using features related to chords to extract features related to harmonic progressions).

An additional 49 features are also proposed for future implementation, including chord-based features [5].

3.3. jWebMiner

jWebMiner [9] is a GUI-based application for extracting features from cultural and demographical information found on the Internet. At its most basic level, the software operates by automatically using Google and Yahoo! web services to acquire statistics on how often particular strings co-occur on the same web pages. This can indicate artist similarity, for example, by measuring how often artists' names co-occur with one another or, to give another

example, to classify artists by genre by measuring how often their names co-occur with particular genre titles.

Search results are processed statistically by jWebMiner in a variety of ways in order to improve results. Further processing options include the abilities to filter out sites containing specified strings, to require that sites contain certain strings in order to be counted and to weight results from different sources differently.

jWebMiner can parse iTunes XML, ACE XML, Weka ARFF [13] or delimited text files in order to conveniently access the particular strings to use in searches.

3.4. ACE

ACE [6] is a meta-learning software package for selecting, optimizing and applying machine learning algorithms. Given a set of extracted features, ACE experiments with a variety of classifier algorithms, parameters, ensemble architectures and dimensionality reduction techniques in order to arrive at a good configuration for the problem at hand. This can be helpful, as a particular algorithm can be more or less appropriate for a given problem in terms of classification accuracy, training speed and classification speed. Even experts in machine learning can have difficulty choosing the best algorithms and parameterizations.

ACE is designed to automate this choice and to facilitate the use of powerful machine learning technology by users of all technical levels, including users with no background in machine learning. ACE also provides a framework for experimenting with new algorithms. ACE is based on the Weka machine learning package [13], so new algorithms developed within the Weka framework can be easily added to ACE. ACE may also be used directly as a classifier.

A significant amount of work has been done on ACE since its original publication. This includes a considerably expanded command-line interface, a partially completed GUI, implementation of a custom cross-validation system that improves on Weka's approach, the provision of additional statistics for use in comparing and evaluating algorithms, and a general restructuring that simplifies the API and makes integration with external software easier.

3.5. ACE XML

ACE XML [6] is a set of standardized file formats for representing feature values extracted from instances, abstract feature descriptions and parameterizations, instance labels and annotations, and class ontologies.

These formats have been designed to address the significant shortcomings in the file formats most commonly used in MIR. To provide just a few examples, ACE XML makes it possible to associate multiple class labels with a single instance, to specify relationships between class labels, to group associated feature values in ways that can be meaningful to machine learning algorithms, and to maintain associations between instances and their sub-sections and metadata, none of which can be done conveniently with the file formats that are most typically used (e.g., Weka ARFF [13]). All of the jMIR components can read and write ACE XML, although they can also be used with Weka ARFF files if necessary.

A major update to ACE XML is in the process of being finalized. The resulting ACE XML 2.0 formats [11] will be proposed as standards for storing MIR-related information and communicating it between software packages in general. Improvements include, to give just a few examples, the abilities to link to external resources using RDF-like triples, to specify weighted class memberships, to express feature arrays of arbitrary dimensionality, to reduce file sizes using compression, etc.

3.6. jMusicMetaManager

jMusicMetaManager [8] is a GUI-based software package for profiling and managing large musical datasets and for detecting metadata errors and inconsistencies in them. These tasks are essential, as the success of ground-truth training and evaluation data is contingent upon the quality of the musical datasets from which they are drawn and the accuracy of the associated metadata.

jMusicMetaManager uses many metrics to find dataset inconsistencies and redundancies. These can be used to detect mislabeled duplicate recordings that could cross-contaminate training and testing sets, for example, or to detect misspellings that might cause "Mingus, Charles" and "Charlie Mingus", for example, to be erroneously treated as two different artists during training and evaluation.

In all, jMusicMetaManager provides users with 23 pre-processing operations, as well as several edit-distance and word ordering/subset operations. A total of 39 different HTML reports can also be automatically generated to help profile and publish musical datasets.

jMusicMetaManager can parse iTunes XML files and MP3 ID3 tags as well as ACE XML and Weka ARFF files in order to access the metadata that is to be analyzed.

3.7. Codaich, Bodhidharma MIDI and SAC

Codaich [8], which has been significantly expanded since its original publication, is an audio dataset consisting of 31,300 MP3 recordings by 2811 artists and belonging to 57 genres of music. These recordings are labelled with 19 metadata fields. The Bodhidharma MIDI dataset [5] is a collection of 950 MIDI recordings belonging to 38 genres. Finally, the SAC dataset [10] consists of matching MP3 recordings, MIDI encodings and cultural data for 250 pieces of music belonging to 10 genres of music.

These datasets have all been used in research with the jMIR components. The SAC dataset in particular is a prototype designed for research on combining audio, symbolic and cultural data.

These datasets are intended to eventually be made publicly accessible using an OMEN-like [8] system. Such systems enable custom feature extraction requests to be processed at distributed sites with legal access to music so that the features can then themselves be distributed elsewhere without violating copyright legislation.

3.8. jMIRUtilities

The previously unpublished jMIRUtilities is a set of tools for performing miscellaneous useful tasks associated with jMIR. These tools include a GUI for batch-associating class labels with instances, utilities for merging various kinds of information, and utilities for extracting information from iTunes XML files or delimited text files.

4. CONCLUSIONS AND FUTURE RESEARCH

jMIR is an open, extensible and easy-to-use suite of software tools that can be used in MIR and automatic music classification research by users of all technical backgrounds. It can be used both directly and as a framework for developing and sharing new approaches.

All of the jMIR components will continue to be expanded and improved by the authors and, it is hoped, the music research community as a whole. The jMIR components provide an infrastructure for such collaborative development, and also provide powerful core libraries of features, machine learning algorithms and musical data that can contribute to the avoidance of duplicated effort.

The heterogeneity of existing MIR tools is an important issue that needs to be addressed, as the availability of too many tools can have the unintended effect of placing barriers between user groups. This issue is one of the central motivators behind the ACE XML 2.0 file formats, as they are designed to enable flexible and expressive data communication across toolsets. Future work will emphasize the provision of developer tools and application-specific plug-ins to facilitate the integration of ACE XML compatibility into other toolsets. Work will also be done on making it even easier to integrate the jMIR components into other software via simple APIs. Another important priority is the development of a common repository for new jMIR modules to be posted and shared.

jMIR and ACE XML are part of the international NEMA (nema.lis.uiuc.edu) project, and the ongoing improvements to jMIR's accessibility are being developed and promoted in this context.

5. ACKNOWLEDGEMENTS

The authors would like to thank the *Andrew W. Mellon Foundation,* the *Social Sciences and Humanities Research Council* (SSHRC), and the *Canadian Foundation for Innovation* (CFI) for their generous financial support. Daniel McEnnis and Jessica Thompson have respectively made substantial contributions to jAudio and to ACE.

6. REFERENCES

[1] Arumi, P., and X. Amatriain. 2005. CLAM, an object oriented framework for audio and music. *Proceedings of the International Linux Audio Conference*

[2] Eerola, T., and P. Toiviainen. 2004. MIR in Matlab: The MIDI Toolbox. *Proceedings of International Conference on Music Information Retrieval*. 22–7.

[3] Lartillot, O., P. Toiviainen, and T. Eerola. 2008. A Matlab toolbox for music information retrieval. In *Data Aalysis, Machine Learning and Applications,* ed. C. Preisach, H. Burkhardt, L. Schmidt-Thieme, and R. Decker. New York: Springer. 261–8.

[4] McEnnis, D., C. McKay, and I. Fujinaga. 2006. jAudio: Additions and improvements. *Proceedings of the International Conference on Music Information Retrieval*. 385–6.

[5] McKay, C. 2004. Automatic genre classification of MIDI recordings. *M.A. Thesis.* McGill University, Canada.

[6] McKay, C., R. Fiebrink, D. McEnnis, B. Li, and I. Fujinaga. 2005. ACE: A framework for optimizing music classification. *Proceedings of the International Conference on Music Information Retrieval*. 42–9.

[7] McKay, C., and I. Fujinaga. 2006. jSymbolic: A feature extractor for MIDI files. *Proceedings of the International Computer Music Conference*. 302–5.

[8] McKay, C., D. McEnnis and I. Fujinaga. 2006. A large publicly accessible prototype audio database for music research. *Proceedings of the International Conference on Music Information Retrieval.* 160–3.

[9] McKay, C., and I. Fujinaga. 2007. jWebMiner: A web-based feature extractor. *Proceedings of the International Conference on Music Information Retrieval.* 113–4.

[10] McKay, C., and I. Fujinaga. 2008. Combining features extracted from audio, symbolic and cultural sources. *Proceedings of the International Conference on Music Information Retrieval.* 597–602.

[11] McKay, C., and I. Fujinaga. 2009. Expressing musical features, class labels, ontologies and metadata using ACE XML 2.0. Submitted to *Structuring Music through Markup Language: Designs and Architectures,* J. Steyn ed. Hershey: IGI.

[12] Tzanetakis, G., and P. Cook. 2000. Marsyas: A framework for audio analysis. *Organized Sound* 10: 293–302.

[13] Witten, I., and E. Frank. 2005. *Data mining: Practical machine learning tools and techniques with Java implementations.* San Francisco: Morgan Kaufmann.