jMIR Overview

Cory McKay

Marianopolis College and CIRMMT

Montreal, Canada

Lecture contents

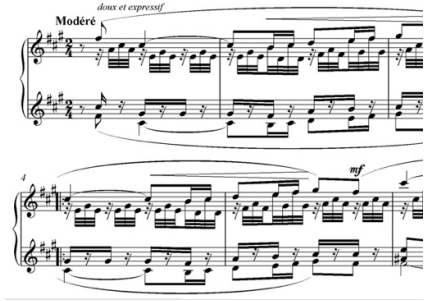
- Introduction to automatic music classification
- Introduction to the jMIR software
- The jMIR components
  - One by one
- Paper ideas
  - Previous collaborations

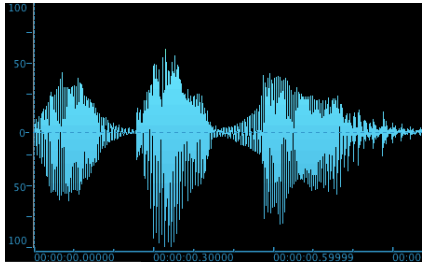
Goal of automatic music classification

- Learn some way of mapping "features" extracted from an "instance" to one or more "classes"
  - Instance: an item to be classified
    - e.g. a song
  - Features: representative information extracted from an instance
    - e.g. amount or chromatic motion in a song
  - Class: a category of interest
    - e.g. a genre, mood, artist, composer, instrument, etc.
- This mapping is typically learned using some form of pattern recognition and machine learning

Main sources of information

- Symbolic recordings
  - e.g. MIDI
- Audio recordings
  - e.g. MP3
- Cultural data
  - e.g. web data, metadata tags, etc.
- Lyrics
- Others
  - Album art, videos, etc.





Automatic music classification

- Typical procedure:
  - Collect annotated training / testing data
    - With appropriate ontologies
  - Extract features
  - Reduce feature dimensionality
  - Train a classification model
    - Typically supervised
  - Validate the model
- Most significant challenges:
  - Acquiring sufficiently large annotated datasets
  - Designing features that encapsulate relevant data

Overview of the jMIR software

- jMIR is software suite designed for performing research in automatic music classification
- Primary tasks performed:
  - Feature extraction
  - Machine learning
  - Data storage file formats
  - Dataset management
    - Acquiring, correcting and organizing metadata

Characteristics of jMIR

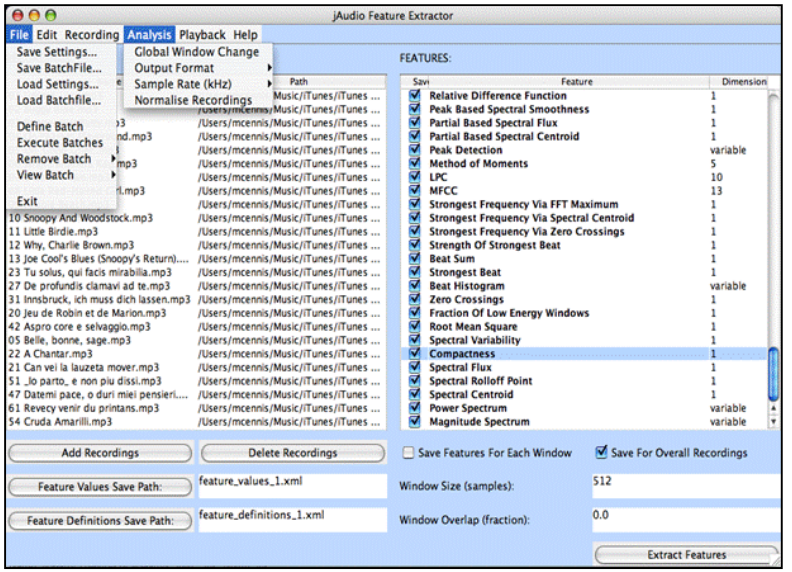
- Has a separate software component to address each important aspect of automatic music classification
  - Each component can be used independently
  - Can also be used as an integrated whole
- Free and open source
- Architectural emphasis on providing an extensible platform for iteratively developing new techniques and algorithms
- Interfaces designed for both technical and non-technical users
- Facilitates multimodal research

jMIR components

- jAudio: Audio feature extraction
- jSymbolic: Feature extraction from MIDI files
- jWebMiner: Cultural feature extraction
- jLyric: Extracts features from lyrical transcriptions
- ACE: Meta-learning classification engine
- ACE XML: File formats
  - Features, feature metadata, instance metadata and ontologies
- lyricFetcher: Lyric mining
- Codaich, Bodhidharma MIDI and SLAC: datasets
- jMusicMetaManager: Metadata management
- jSongMiner: Metadata harvesting
- jMIRUtilities: Infrastructure for conducting experiments

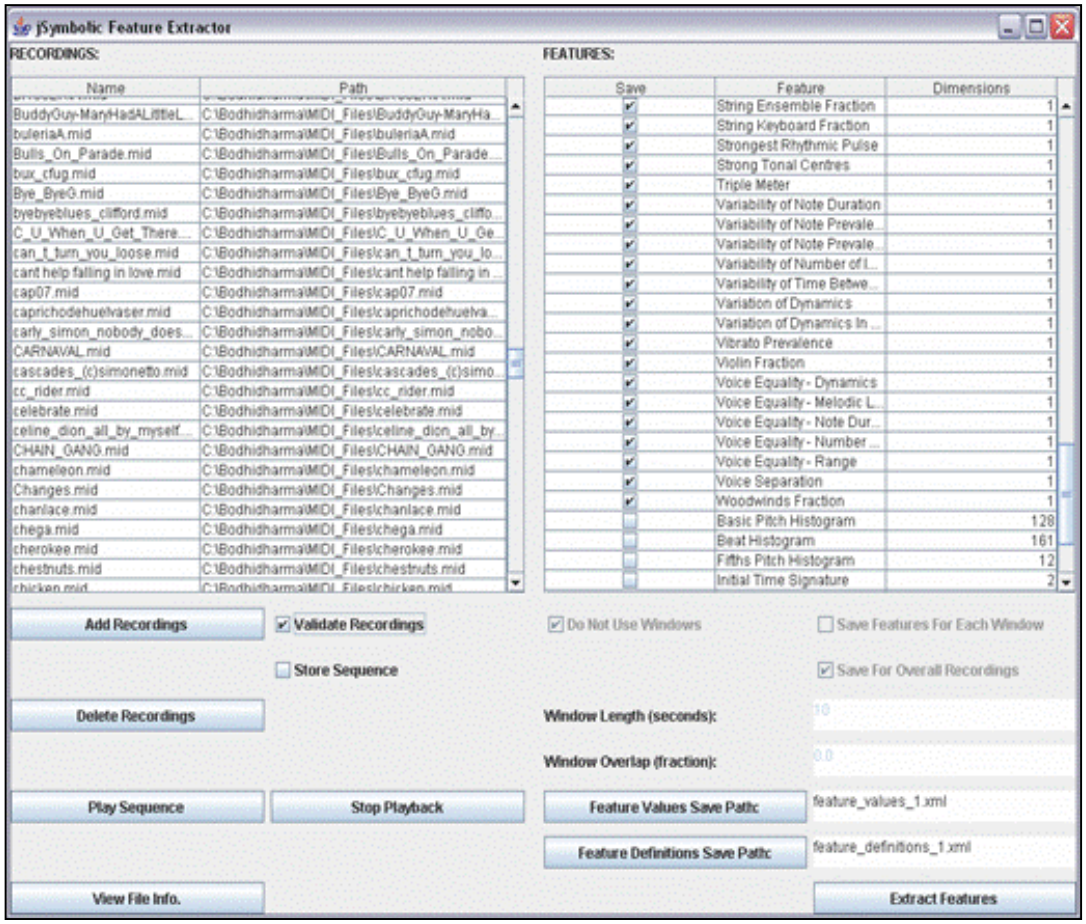
jAudio: Audio feature extractor

- Implemented jointly with Daniel McEnnis
- Extracts features from audio files
  - MP3, WAV, AIFF, AU, SND
- 28 bundled core features
  - Mainly low-level, some high-level
- Can automatically generate new features using metafeatures and aggregators
  - e.g. the change in a feature value from window to window
- Includes tools for testing new features being developed
  - Synthesize audio, record audio, sonify MIDI, display audio, etc.



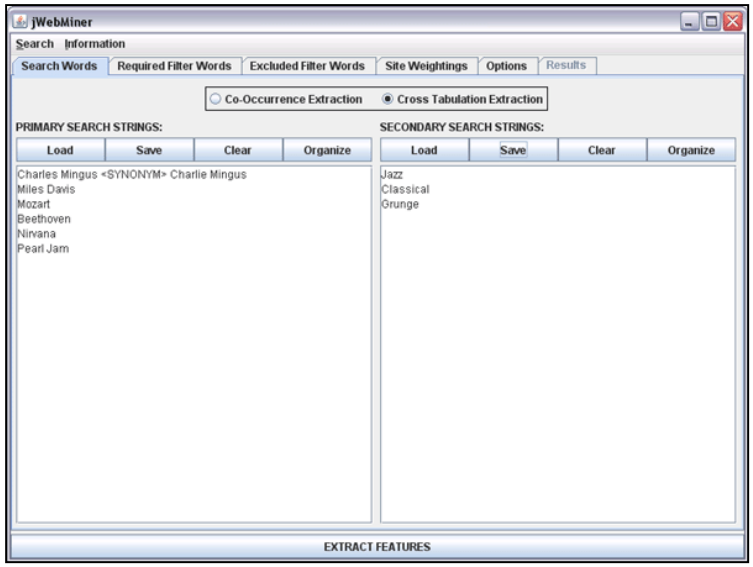
jSymbolic: Symbolic feature extractor

- Extracts features from MIDI files
- 111 implemented features
  - By far the largest existing symbolic feature catalogue
  - Many are original
- An additional 49 features are proposed but not yet implemented
- Features saved to ACE XML



jWebMiner: Cultural feature extractor

- Extracts cultural features from the web using search engine web services
- Calculates how often particular strings co-occur on the same web pages
  - e.g. how often does “J. S. Bach” co-occur on a web page with “Baroque”, compared to “Prokofiev”?
  - Results are processed to remove noise
- Additional options:
  - Can assign weights to particular sites
  - Can enforce filter words
  - Permits synonyms
- Also calculates features based on Last.FM user tags frequencies



lyricFetcher: Lyric miner

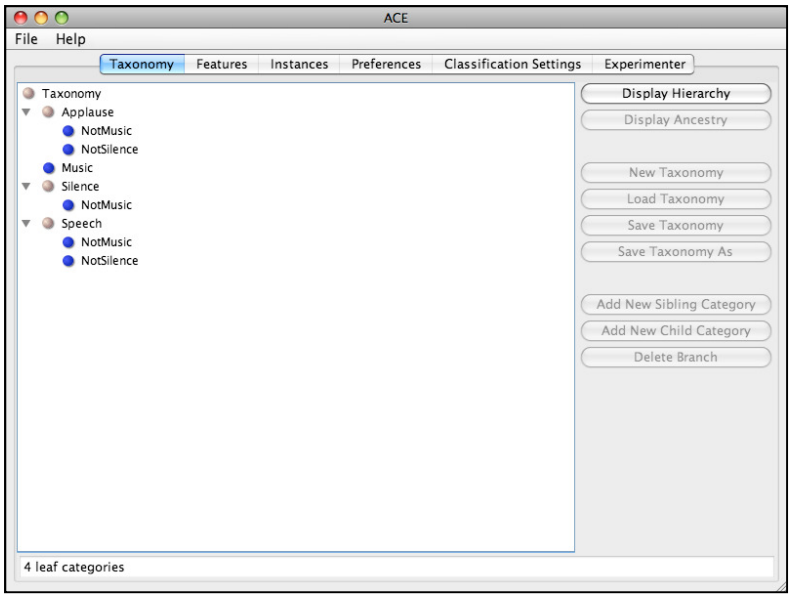
- lyricFetcher automatically harvests lyrics from on-line lyrics repositories
  - LyricWiki and LyricsFly
  - Queries based on lists of song titles and artist names
- Post-processing is applied to the lyrics in order to make remove noise and make them sufficiently consistent for feature extraction
  - Deals with situations where sections of lyrics are abridged using keywords such as “chorus”, “bridge”, “verse”, etc.
  - Filters out keywords that could contaminate the lyrics
- Ruby implementation

jLyrics: Lyrical feature extractor

- Extracts features from lyrics stored in text files:
  - Automated Readability Index
  - Average Syllable Count Per Word
  - Contains Words
  - Flesh-Kincaid Grade Level
  - Flesh Reading Ease
  - Function Word Frequencies
  - Letter-Bigram Components
  - Letter Frequencies
  - Letters Per Word Average
  - Letters Per Word Variance
  - Lines Per Segment Average
  - Lines Per Segment Variance
  - Number of Lines
  - Number of Segments
  - Number of Words
  - Part-of-Speech Frequencies
  - Punctuation Frequencies
  - Rate of Misspelling
  - Sentence Count
  - Sentence Length Average
  - Topic Membership Probabilities
  - Vocabulary Richness
  - Vocabulary Size
  - Word Profile Match
  - Words Per Line Average
  - Words Per Line Variance
- Can also automatically generate word frequency profiles for particular classes if training data is provided
- Central framework implemented in Java
  - Other technologies used by third-party components

ACE: Meta-learning engine

- Evaluates the relative suitability of different dimensionality reduction and classification algorithms for a given problem
  - Can also train and classify with manually selected algorithms
- Evaluates algorithms in terms of
  - Classification accuracy
  - Consistency
  - Time complexity
- Based on the Weka framework, so new algorithms can be added easily



ACE XML: MIR research file formats

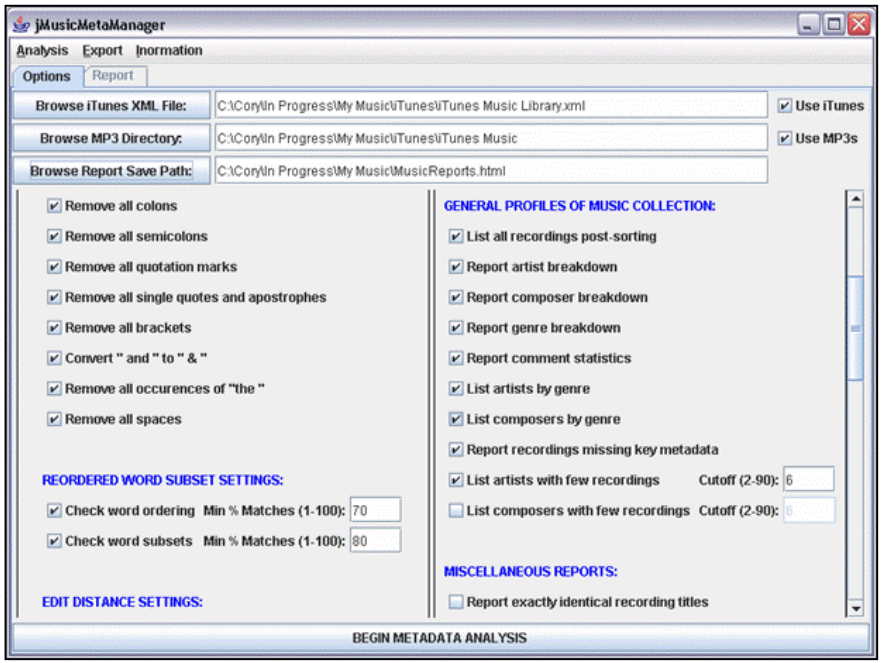
- Standardized file formats that can represent:
  - Feature values extracted from instances
  - Abstract feature descriptions and parameterizations
  - Instance labels and annotations
  - Class ontologies
- Designed to be flexible and extensible
  - Able to express types of information that are particularly pertinent to music
- Allow jMIR components to communicate with each other
  - Can also be adopted for independent use by other software
- ACE XML 2.0 provides even more expressivity
  - e.g. potential for integration into RDF ontologies

jMIR datasets

- Codaich is a MP3 research set
  - Carefully cleaned and labelled
  - The published 2006 version has 26,420 recordings
    - Belonging to 55 genres
    - Is constantly growing: currently 35,363 MP3s
- Bodhidharma MIDI has 950 MIDI recordings
  - 38 genres of music
- SLAC consists of 250 matched audio recordings, MIDI recordings, lyrical transcriptions and metadata that can be used to extract cultural features
  - Useful for experiments on combining features from different types of data
  - 10 genres of music (in 5 pairs of similar genres)

jMusicMetaManager: Dataset manager

- Detects metadata errors/inconsistencies and redundant copies of recordings
- Detects differing metadata values that should in fact be the same
  - e.g. “Charlie Mingus” vs. “Mingus, Charles”
- Generates HTML inventory and profile reports (39 reports in all)
- Parses metadata from ID3 tags and iTunes XML



jSongMiner: Metadata miner

- Software for automatically acquiring formatted metadata about songs, artists and albums
- Designed for use with the Greenstone digital library software
  - May also be used for other purposes, such as cultural feature extraction
- Identifies music files
  - Uses Echo Nest fingerprinting functionality and embedded metadata
- Mines a wide range of metadata tags from the Internet and collates them in a standardized way
  - Data extracted from The Echo Nest, Last.FM, MusicBrainz, etc.
  - Over 100 different fields are extracted
  - Data may be formatted into unqualified and/or qualified Dublin Core fields if desired
- Saves the results in ACE XML or text
  - Can also be integrated automatically into a Greenstone collection

Previous collaborations (1/2)

- Vigliensoni, McKay and Fujinaga (2010)
  - Addition of Last.FM functionality to jWebMiner
  - Empirical comparison of different kinds of cultural data
- Angeles, McKay and Fujinaga (2010)
  - Addition of MusicBrainz functionality to jMusicMetaManager
  - Empirical comparison of curated, noisy and automatically cleaned metadata
- McKay, Burgoyne, Hockman, Smith, Vigliensoni and Fujinaga (2010)
  - Development of jLyrics and lyricFetcher
  - Empirical comparison of different feature types
- Thompson, McKay, Burgoyne and Fujinaga (2009)
  - Improvements to ACE

Previous collaborations (2/2)

- McKay, Burgoyne, Thompson and Fujinaga (2009)
  - Improvements to ACE XML
- McEnnis, McKay and Fujinaga (2006)
  - Improvements to jAudio
- Fiebrink, McKay and Fujinaga (2005)
  - An empirical investigation of dimensionality reduction using ACE (and other technologies)
- Sinyor, McKay, Fiebrink, McEnnis and Fujinaga (2005)
  - Beatboxing classification using ACE and jAudio

Paper ideas: Improve components

- jSongMiner
  - Add more data sources
- jAudio
  - Develop more features, especially psychologically meaningful features
  - Improve interface
- jSymbolic
  - Develop more features or implement the remaining feature catalogue
- jWebminer
  - Take advantage of additional web services (e.g. Amazon) to add more features
- jLyrics
  - Develop features especially relevant to music
- ACE
  - Add more machine learning algorithms
  - Especially unsupervised learning
- jMusicMetaManager
  - Add correction functionality

Paper ideas: Develop new components

- OMEN
  - Reimplement and get working
- jImage
  - Extract features from album art, press photos, etc.
- jVideo
  - Extract features from music videos, concert videos, etc.
- jMusicVisualiser
  - Offer visual ways of exploring relationships between musical instances, features and classes
- jStructure:
  - Automatically segment audio streams in time, both in terms of partitioning separate pieces of music within a single stream and in terms of structural divisions within individual pieces
- jClassOntology
  - Use data mining to harvest class ontologies

Paper ideas: Apply jMIR

- There are many problems to which the jMIR components could be applied
  - Either directly or with specialized improvements (e.g. features developed especially for chord recognition)
- Consider the many MIREX application areas
  - www.music-ir.org/mirex/wiki/MIREX_HOME
- More ideas:
  - See the Future Research sections of my papers and dissertation

More information

- Overview, documentation and publications
  - jmir.sourceforge.net
- Code and (in most cases) manuals
  - sourceforge.net/projects/jmir/
  - sourceforge.net/projects/jaudio/
- My dissertation on jMIR
  - www.music.mcgill.ca/~cmckay/papers/musictech/mckay10dissertation.pdf
- E-mail
  - cory.mckay@mail.mcgill.ca

Acknowledgements

- Collaborators at McGill and Waikato Universities
- Funding, past and present:
  - CIRMMT
  - Centre for Open Software Innovation
  - Andrew W. Mellon Foundation
  - Social Sciences and Humanities Research Council of Canada
  - Canadian Foundation for Innovation

- Contact information:
  - jmir.sourceforge.net
  - cory.mckay@mail.mcgill.ca