JSYMBOLIC 2.2: EXTRACTING FEATURES FROM SYMBOLIC MUSIC FOR USE IN MUSICOLOGICAL AND MIR RESEARCH

Cory McKay
Marianopolis College
cory.mckay@mail.mcgill.ca

Julie E. Cumming
McGill University
julie.cumming@mcgill.ca

Ichiro Fujinaga
McGill University
ichiro.fujinaga@mcgill.ca

ABSTRACT

jSymbolic is an open-source platform for extracting features from symbolic music. These features can serve as inputs to machine learning algorithms, or they can be analyzed statistically to derive musicological insights.

jSymbolic implements 246 unique features, comprising 1497 different values, making it by far the most extensive symbolic feature extractor to date. These features are designed to be applicable to a diverse range of musics, and may be extracted from both symbolic music files as a whole and from windowed subsets of them. Researchers can also use jSymbolic as a platform for developing and distributing their own bespoke features, as it has an easily extensible plug-in architecture.

In addition to implementing 135 new unique features, version 2.2 of jSymbolic places a special focus on functionality for avoiding biases associated with how symbolic music is encoded. In addition, new interface elements and documentation improve convenience, ease-of-use and accessibility to researchers with diverse ranges of technical expertise. jSymbolic now includes a GUI, command-line interface, API , flexible configuration file format, extensive manual and detailed tutorial.

The enhanced effectiveness of jSymbolic 2.2's features is demonstrated in two sets of experiments: 1) genre classification and 2) Renaissance composer attribution.

1. INTRODUCTION

The majority of research performed by musicologists, music theorists, music librarians and others focuses on symbolic music representations. Unfortunately, relatively few MIR-oriented software tools are available to assist such research, particularly with respect to research involving the increasingly large corpora being studied.

jSymbolic is an open-source Java framework designed to at least partially address this shortcoming. Its primary function is to extract a large number of features (statistical descriptors) from potentially huge collections of digitally-represented symbolic music. These features can then be used to directly assist music researchers in analysis and search-based tasks, as well as in research incorporating machine learning.

Possible research applications include: empirical testing of existing musicological theories [11]; exploratory research that can reveal unexpected insights [11]; reconciling historical evidence with content-based data [12]; annotation of large corpora to allow content-based searches [10]; performing multimodal research by combining symbolic features with audio, textual and other features [9]; and generating novel music in specific styles by using feature values as stylistic guideposts [23].

jSymbolic 2.2 has been dramatically improved and expanded since its last properly published version (1.2) was released in 2010 [9]. It is also a component of the larger jMIR research software framework [9]. jSymbolic and the other jMIR components (including source code) can all be downloaded from [13].

2. RELATED RESEARCH

Surprisingly few frameworks designed specifically for extracting features from symbolic music have been published, although there are several MIR toolkits for analyzing symbolic music more generally. The MIDI Toolbox [6] is one particularly well-known system implemented in Matlab. The powerful music21 analysis toolkit [4] includes ports of 57 of the original jSymbolic 1.2 features, and also offers substantial additional useful functionality.

The Humdrum toolkit [8] is a well-known tool for analyzing music, although it does not extract features as such. The Melisma Music Analyzer [22] is another excellent analysis-oriented system, and `pretty_midi` [17] provides helpful creation, manipulation and extraction tools.

Additional work has been published where symbolic feature extraction is performed as a part of larger research projects, but where the feature extraction code has not itself been published. Standouts include [1] and [15]. Corrêa and Rodrigues have written a nice survey of related symbolic genre classification research [2].

To the best of our knowledge, there is no existing software that extracts anywhere near the number or diversity of features as jSymbolic, nor is there any with the same focus on broad accessibility and extensibility.

3. CHARACTERISTICS OF JSYMBOLIC

3.1 Features Extracted

jSymbolic 2.2 extracts 246 unique features, some of which are multidimensional, for a total of 1497 values (version 1.2 extracted 111 features and 1022 values). Details on the original musicological and music theoretical sources and motivations for the features are available in [9]. The features can be divided into eight general groups:

- **Pitch Statistics:** How common are various pitches and pitch classes relative to one another? How are they distributed and how much do they vary?
- **Melodic Intervals:** What melodic intervals are present? How much melodic variation is there? What can be observed from melodic contour measurements?
- **Chords and Vertical Intervals:** What vertical intervals are present? What types of chords do they represent? What kinds of harmonic movement are present?
- **Rhythm:** Information associated with note attacks, durations and rests, measured in ways that are both dependent and independent of tempo. Information on rhythmic variability, including rubato, and meter.
- **Instrumentation:** Which instruments are present, and which are emphasized relative to others? Both pitched and non-pitched instruments are considered.
- **Texture:** How many independent voices are there and how do they interact (e.g. parallel vs. contrary motion)? What is the relative importance of voices?
- **Dynamics:** How loud are notes and what kinds of variations in dynamics occur?
- **MEI-Specific:** Information that cannot be represented explicitly in MIDI (e.g. slurs) but can be in the Music Encoding Initiative (MEI) file format [16].

See Figure 1 for a complete list of the jSymbolic 2.2 features, including indications of which ones are new, as well as which ones are multidimensional.

These features are designed to be wide-ranging, in order to be applicable to a diverse range of musics from a variety of cultures, styles and time periods. A few features are intentionally partially redundant; for example, the Vertical Interval Histogram indicates the number of minor thirds and major thirds (among other things) separately, but the Vertical Thirds feature combines them. Such partial redundancies help highlight patterns in alternative ways to musicologists examining features. Also, some features are based on information explicitly (but not necessarily correctly) specified as metadata in the input files, such as meter or key, and others attempt to infer such information directly from the music itself.

3.2 Designing New Features

Extensibility and modularity are key priorities, as jSymbolic is intended to be a platform for developing and testing new features just as much as it is an out-of-the-box tool. New features can be added as plug-ins simply by extending an existing Java class, and it is easy to incorporate the values of existing features into new features in order to iteratively build new features of increasing sophistication. jSymbolic also automatically handles all infrastructure relating to feature dependencies and extraction scheduling. The overall design of the software is extensible, as is its configuration file format.

A tool has been added for exploring MIDI messages directly at a low-level, in order to help debug new features. jSymbolic also now automatically validates and error-checks new features as they are added, and there is substantial new general unit testing infrastructure.

3.3 Configuration Files

jSymbolic now includes a flexible configuration file format that can be used for batch processing, as a way of applying consistent settings across sessions and for keeping a record of settings used in individual experiments. These configuration files can be saved with the GUI, or they can be edited directly.

3.4 Avoiding Systematic Encoding Bias

One must always be careful that extracted features are not correlated with the source of data rather than its underlying musical content. This could happen, for example, in a corpus constructed by joining data from different sources, where each source uses different encoding conventions (e.g. different instrumentation designations for voices, or different interpretations of tempo markings). Such issues have been discussed regarding audio [21], but less so for symbolic data. Ideally, all data in a corpus would be systematically encoded in the same way, but this is rarely the case in practice.

jSymbolic therefore now includes functionality for generating "consistency reports." These automatically check sets of symbolic music files for such biases.

An optional "safe" configuration file is also provided, which disables features associated with instrumentation, dynamics, microtonal pitches and tempo, as these tend to be particularly vulnerable to encoding bias. This is especially useful for musics where these qualities are typically unspecified, such as Renaissance music.

Many of jSymbolic 2.2's new features are also specifically designed to avoid such biases. For example, many of the new rhythmic features are tempo-independent, so that they can be used even if tempo is source-correlated, while the old tempo-linked features can still be used if tempos are meaningfully and consistently encoded.

[3] presents a more detailed analysis of related issues, including empirical results produced with jSymbolic 2.2.

3.5 Windowed Extraction

Users can now perform windowed feature extraction with jSymbolic, with overlapping or non-overlapping windows, as well as extraction over entire pieces. Although common with audio, this ability to extract features separately from subsets of a piece is rare in the symbolic domain, and enables powerful new kinds of analysis.

Overall Pitch Statistics
(128) Basic Pitch Histogram
(12) Pitch Class Histogram
(12) Folded Fifths Pitch Class Histogram
Number of Pitches
Number of Pitch Classes
Number of Common Pitches
Number of Common Pitch Classes
Range
Importance of Bass Register
Importance of Middle Register
Importance of High Register
Dominant Spread
Strong Tonal Centres
Mean Pitch
Mean Pitch Class
Most Common Pitch
Most Common Pitch Class
Prevalence of Most Common Pitch
Prevalence of Most Common Pitch Class
Relative Prevalence of Top Pitches
Relative Prevalence of Top Pitch Classes
Interval Between Most Prevalent Pitches
Interval Between Most Prevalent Pitch Classes
Pitch Variability
Pitch Class Variability
Pitch Class Variability After Folding
Pitch Skewness
Pitch Class Skewness
Pitch Class Skewness After Folding
Pitch Kurtosis
Pitch Class Kurtosis
Pitch Class Kurtosis After Folding
Major or Minor
First Pitch
First Pitch Class
Last Pitch
Last Pitch Class
Glissando Prevalence
Average Range of Glissandos
Vibrato Prevalence
Microtone Prevalence

Melodic Intervals
(128) Melodic Interval Histogram
Most Common Melodic Interval
Mean Melodic Interval
Number of Common Melodic Intervals
Distance Between Most Prevalent Melodic Intervals
Prevalence of Most Common Melodic Interval
Relative Prevalence of Most Common Melodic Intervals
Amount of Arpeggiation
Repeated Notes
Chromatic Motion
Stepwise Motion
Melodic Thirds
Melodic Perfect Fourths
Melodic Tritones
Melodic Perfect Fifths
Melodic Sixths
Melodic Sevenths
Melodic Octaves
Melodic Large Intervals
Minor Major Melodic Third Ratio
Melodic Embellishments
Direction of Melodic Motion
Average Length of Melodic Arcs
Average Interval Spanned by Melodic Arcs
Melodic Pitch Variety

Chords and Vertical Intervals
(128) Vertical Interval Histogram
(12) Wrapped Vertical Interval Histogram
(11) Chord Type Histogram
Average Number of Simultaneous Pitch Classes
Variability of Number of Simultaneous Pitch Classes
Average Number of Simultaneous Pitches
Variability of Number of Simultaneous Pitches
Most Common Vertical Interval
Second Most Common Vertical Interval
Distance Between Two Most Common Vertical Intervals
Prevalence of Most Common Vertical Interval
Prevalence of Second Most Common Vertical Interval
Prevalence Ratio of Two Most Common Vertical Intervals
Vertical Unisons
Vertical Minor Seconds
Vertical Thirds
Vertical Tritones
Vertical Perfect Fourths
Vertical Perfect Fifths
Vertical Sixths
Vertical Sevenths
Vertical Octaves
Perfect Vertical Intervals
Vertical Dissonance Ratio
Vertical Minor Third Prevalence
Vertical Major Third Prevalence
Chord Duration
Partial Chords
Standard Triads
Diminished and Augmented Triads
Dominant Seventh Chords
Seventh Chords
Non-Standard Chords
Complex Chords
Minor Major Triad Ratio

Rhythm
(2) Initial Time Signature
Simple Initial Meter
Compound Initial Meter
Complex Initial Meter
Duple Initial Meter
Triple Initial Meter
Quadruple Initial Meter
Metrical Diversity
Total Number of Notes
Note Density per Quarter Note
Note Density per Quarter Note per Voice
Note Density per Quarter Note Variability
(12) Rhythmic Value Histogram
Range of Rhythmic Values
Number of Different Rhythmic Values Present
Number of Common Rhythmic Values Present
Prevalence of Very Short Rhythmic Values
Prevalence of Short Rhythmic Values
Prevalence of Medium Rhythmic Values
Prevalence of Long Rhythmic Values
Prevalence of Very Long Rhythmic Values
Prevalence of Dotted Notes
Shortest Rhythmic Value
Longest Rhythmic Value
Mean Rhythmic Value
Most Common Rhythmic Value
Prevalence of Most Common Rhythmic Value
Relative Prevalence of Most Common Rhythmic Values
Difference Between Most Common Rhythmic Values
Rhythmic Value Variability
Rhythmic Value Skewness
Rhythmic Value Kurtosis
(12) Rhythmic Value Median Run Lengths Histogram
Mean Rhythmic Value Run Length
Median Rhythmic Value Run Length
Variability in Rhythmic Value Run Lengths
(12) Rhythmic Value Variability in Run Lengths Histogram
Mean Rhythmic Value Offset
Median Rhythmic Value Offset
Variability of Rhythmic Value Offsets
Complete Rests Fraction
Partial Rests Fraction
Average Rest Fraction Across Voices
Longest Complete Rest
Longest Partial Rest
Mean Complete Rest Duration
Mean Partial Rest Duration
Median Complete Rest Duration
Median Partial Rest Duration
Variability of Complete Rest Durations
Variability of Partial Rest Durations
Variability Across Voices of Combined Rests
(161) Beat Histogram Tempo Standardized
Number of Strong Rhythmic Pulses - Tempo Standardized
Number of Moderate Rhythmic Pulses - Tempo Standardized
Num. Relatively Strong Rhythmic Pulses - Tempo Standardized
Strongest Rhythmic Pulse - Tempo Standardized
Second Strongest Rhythmic Pulse - Tempo Standardized
Harmonicity of Two Strongest Rhythmic Pulses - Tempo Stand.
Strength of Strongest Rhythmic Pulse - Tempo Standardized
Strength of Second Strongest Rhythmic Pulse - Tempo Stand.
Strength Ratio of Two Strongest Rhythmic Pulses - Tempo Stand.
Combined Strength of 2 Strongest Rhyth. Pulses - Tempo Stand.
Rhythmic Variability - Tempo Standardized
Rhythmic Looseness - Tempo Standardized
Polyrhythms - Tempo Standardized
Initial Tempo
Mean Tempo
Tempo Variability
Duration in Seconds
Note Density
Note Density Variability
Average Time Between Attacks
Average Time Between Attacks for Each Voice
Variability of Time Between Attacks
Average Variability of Time Between Attacks for Each Voice
Minimum Note Duration
Maximum Note Duration
Average Note Duration
Variability of Note Durations
Amount of Staccato
(161) Beat Histogram
Number of Strong Rhythmic Pulses
Number of Moderate Rhythmic Pulses
Number of Relatively Strong Rhythmic Pulses
Strongest Rhythmic Pulse
Second Strongest Rhythmic Pulse
Harmonicity of Two Strongest Rhythmic Pulses
Strength of Strongest Rhythmic Pulse
Strength of Second Strongest Rhythmic Pulse
Strength Ratio of Two Strongest Rhythmic Pulses
Combined Strength of Two Strongest Rhythmic Pulses
Rhythmic Variability
Rhythmic Looseness
Polyrhythms

Instrumentation
(128) Pitched Instruments Present
(47) Unpitched Instruments Present
(128) Note Prevalence of Pitched Instruments
(47) Note Prevalence of Unpitched Instruments
(128) Time Prevalence of Pitched Instruments
Variability of Note Prevalence of Pitched Instruments
Variability of Note Prevalence of Unpitched Instruments
Number of Pitched Instruments
Number of Unpitched Instruments
Unpitched Percussion Instrument Prevalence
String Keyboard Prevalence
Acoustic Guitar Prevalence
Electric Guitar Prevalence
Violin Prevalence
Saxophone Prevalence
Brass Prevalence
Woodwinds Prevalence
Orchestral Strings Prevalence
String Ensemble Prevalence
Electric Instrument Prevalence

Texture
Maximum Number of Independent Voices
Average Number of Independent Voices
Variability of Number of Independent Voices
Voice Equality - Number of Notes
Voice Equality - Note Duration
Voice Equality - Dynamics
Voice Equality - Melodic Leaps
Voice Equality - Range
Importance of Loudest Voice
Relative Range of Loudest Voice
Relative Range Isolation of Loudest Voice
Relative Range of Highest Line
Relative Note Density of Highest Line
Relative Note Durations of Lowest Line
Relative Size of Melodic Intervals in Lowest Line
Voice Overlap
Voice Separation
Variability of Voice Separation
Parallel Motion
Similar Motion
Contrary Motion
Oblique Motion
Parallel Fifths
Parallel Octaves

Dynamics
Dynamic Range
Variation of Dynamics
Variation of Dynamics in Each Voice
Average Note to Note Change in Dynamics

MEI-Specific
Number of Grace Notes
Number of Slurs

Figure 1. All features implemented by jSymbolic 2.2. Headings in bold refer to feature groups, not features. Features in italics are new (added since jSymbolic 1.2). Numbers in parentheses indicate the size of multi-dimensional features. Detailed descriptions of individual features are available in jSymbolic's manual [14] and in its GUI.

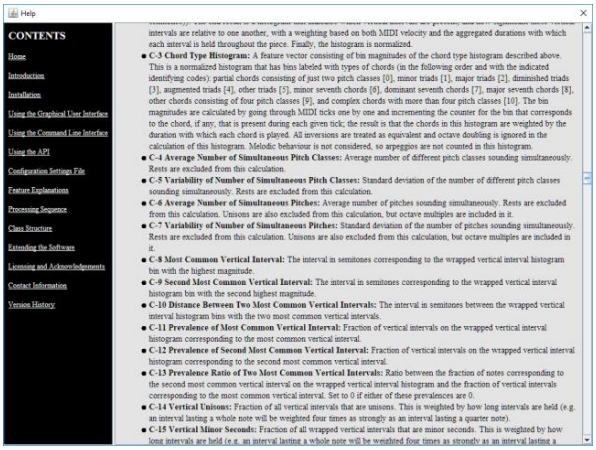

Figure 2. The new jSymbolic 2.2 manual.

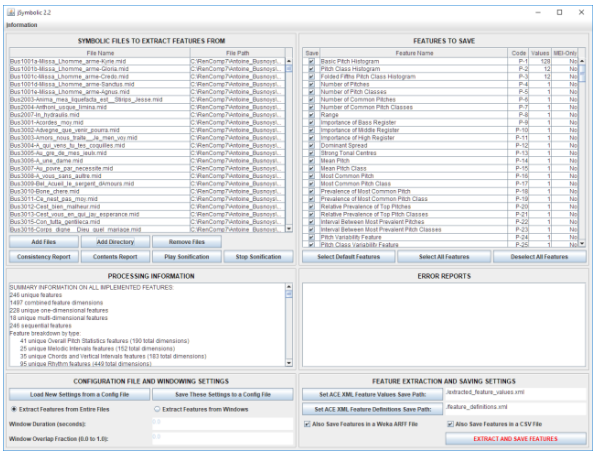

Figure 3. The redesigned jSymbolic 2.2 GUI.

3.6 I/O Formats and jMei2Midi

jSymbolic can extract features from music stored in either MIDI or MEI [16]. MIDI, despite its well-documented limitations, has the essential advantage that it can be read or generated by almost any symbolic music software, and also permits live performance encoding. This latter benefit makes MIDI compatible with non-Western (and Western) musics that do not conform to the quantized tunings and rhythms typical of common practice music notation and the symbolic music formats based on it. MIDI is also more suitable for transcribing audio, which also rarely conforms to strict quantization.

MEI, in turn, is a rich and extensible format that allows many kinds of important information to be represented that cannot be encapsulated with MIDI. jSymbolic's new support for MEI is achieved via our custom-built Java MEI parser and MEI-to-MIDI converter called jMei2Midi, which can also be used as a standalone software library. jMei2Midi performs a more extensive level of MEI conversion than any other converter, and also maintains a channel for preserving and transmitting information that cannot be represented in MIDI.

Although jSymbolic cannot yet directly parse formats such as Music XML, OSC, Humdrum **kern or LilyPond, there are fortunately many converters that can translate such formats to MIDI or MEI for jSymbolic feature extraction. jSymbolic's Rodan [7] wrapper can already do this with Music XML.

Extracted features can now be saved as both Weka ARFF [24] files (a machine learning format) and as general-purpose CSV files. Previously, ACE XML [9] was the only output file format option.

3.7 Usability and Interfaces

It is crucial that jSymbolic be easy to learn and use for users with diverse technical backgrounds, and that it be easily adaptable to a broad range of use cases. This has been a primary focus of the upgrades since version 1.2.

jSymbolic now includes a detailed HTML manual (Figure 2) [14] and an extensive step-by-step tutorial that includes worked exercises with both jSymbolic and the Weka data mining framework [24]. jSymbolic's Java implementation and lack of external dependencies make the software platform-independent and easy-to-install.

The original jSymbolic was only usable via a GUI, which has been substantially improved in version 2.2 (Figure 3). jSymbolic also now also includes command-line interface for batch processing, a well-documented Java API for programmatic access and a Rodan [7] workflow for those wishing to take advantage of distributed processing. New feedback on progress is provided as processing proceeds, and cleaner error handling and more detailed reporting in general have been instituted.

4. GENRE CLASSIFICATION EXPERIMENTS

4.1 Experimental Goals and Methodology

Our first set of experiments involved using the jSymbolic features to classify music by genre. This was done using the MIDI portion of our (balanced) "SAC" dataset [9], which consists of 250 pieces of music. SAC is divided into ten genres: Hardcore Rap, Pop Rap, Bop, Swing, Baroque, Romantic, Alternative Rock, Metal, Modern Blues and Traditional Blues. These genres can be combined pairwise into five parent genres: Rap, Jazz, Classical, Rock and Blues. This ontological structure permits one to evaluate how well a given approach can distinguish between both dissimilar genres (the five parent genres) and similar genres (the two classes comprising each pair).

Features were extracted from SAC using both the old jSymbolic 1.2 [9] and the new jSymbolic 2.2, in order to explore the effects of the new features. All implemented features were used, as no systematic encoding biases were found in the data (see Section 3.4).

The Weka machine learning framework [24] was used to perform 10-fold cross-validation experiments using its SMO support vector machine implementation (with default hyper-parameter settings). No dimensionality reduction pre-processing was applied, beyond what SMO does itself. This simple and generic classification methodology was chosen intentionally, as an important goal of this pa-

per is to emphasize the accessibility of jSymbolic's features to music researchers who may have little or no background in machine learning. A more sophisticated approach would have been used if this were a paper specifically on classification.

4.2 Classification Results and Discussion

Corpus	jSymbolic 1.2		jSymbolic 2.2	
	Accuracy	F-score	Accuracy	F-score
SAC 5	90.4%	0.809	93.2%	0.872
SAC 10	75.6%	0.703	77.6%	0.631

Table 1. SAC (5-class and 10-class) genre classification accuracies and F-scores (averaged across 10 folds).

jSymbolic's performance (Table 1) is quite impressive overall, especially since such basic machine learning techniques were used. Although there has not been a symbolic genre classification MIREX event in over a decade, the 2017 audio genre classification results [5] provide a rough general context: the highest classification accuracies were 75.9% in the 10-class Latin genre task, 76.8% in the 10-class popular genre task and 67.9% in the 7-class K-Pop genre task.

The new 2.2 features provided better classification accuracies than the old 1.2 features on both versions of SAC, by 2.8% and 2.0%. The F-score, however, declined for SAC 10, but improved for SAC 5.

The value of jSymbolic 2.2's greatly expanded feature catalogue has a scope well beyond its classification performance gains. Many music researchers are interested in specifically what it is that differentiates various kinds of music, and a greater number of features make it possible to explore and understand music more thoroughly and precisely. This is revisited below.

5. COMPOSER ATTRIBUTION EXPERIMENTS

5.1 Experimental Goals and Methodology

The second set of experiments involved the same Weka-based classification methodologies described in Section 4.1. This time, however, the experiments involved Renaissance composer attribution; this is much more than a toy problem in early music studies, as there are many pieces whose composer is unknown or disputed, and feature-based machine learning holds significant potential for resolving such debates.

We constructed our (unbalanced) "RenComp7" dataset by combining the Josquin (top two Rodin security levels [19] only, based on historical sources), La Rue, Ockeghem, Busnoys and Martini data from [19] with John Miller's Palestrina data and the Victoria data used in [20]. All files not already encoded as MIDI were converted. The resultant RenComp7 corpus consists of 1584 pieces.

An analysis of the data found that certain features were influenced by systematic encoding bias (see Section 3.4), namely those based on instrumentation, dynamics and tempo. Since including these features would have artificially inflated performance, it was necessary to exclude certain features from consideration. As a result, only 335 of 1022 jSymbolic 1.2 feature values and 801 of 1497 jSymbolic 2.2 feature values were used.

We conducted one experiment where classification was performed among all seven RenComp7 composers. This was followed by two pairwise classifications that are of particular musicological interest: Josquin vs. La Rue (exact contemporaries who are musically similar) and Josquin vs. Ockeghem (from different generations).

5.2 Classification Results and Discussion

Corpus	jSymbolic 1.2		jSymbolic 2.2	
	Accuracy	F-score	Accuracy	F-score
All 7 Composers	87.9%	0.634	92.4%	0.715
Josq / Ockeghem	84.7%	0.818	92.6%	0.911
Josquin / La Rue	82.0%	0.771	86.3%	0.824

Table 2. RenComp7 composer attribution classification accuracies and F-scores (averaged across 10 folds).

The overall ability of the jSymbolic 2.2 features to distinguish between the composers (Table 2) is unprecedented in the automatic classification literature, and is all the more impressive given the simple machine learning methodology used. The excellent work of Brinkman et al. [1] provides the best basis for comparison: the authors used 53 features to classify 6 composers (J. S. Bach and five Renaissance composers), and obtained success rates of roughly 63% on average. Their approach did very well at discriminating Bach from the Renaissance composers (97%). This highlights both the quality of their approach and the particular difficulty of identifying Renaissance composers, and makes the success of the jSymbolic features on exclusively Renaissance music all the more encouraging. The new 2.2 features outperformed the old 1.2 features in all cases.

5.3 Diving into Features

As noted above, the relative performance of individual features can be at least as important in revealing musicological insights as overall classification performance. As an example of research along these lines, we asked two experts on Renaissance music, Julie E. Cumming and Peter Schubert, to predict what characteristics they thought would best differentiate the music of Josquin and Ockeghem, based on their extensive general experience studying the music of the two composers, and without any *a priori* exposure to the feature data. The jSymbolic feature data was then used to test these expectations. The results, as outlined in Figure 4, demonstrate how some of their predictions were indeed confirmed, but others were shown to be incorrect. This emphasizes the general need in musicology and music theory for empirical validation of a wide range of widespread beliefs and assumptions that have never been confirmed via systematic studies of large datasets. It is hoped that jSymbolic and similar software can help address this issue.

Empirical Testing of Expert Predictions of Characteristics More Evident in Ockeghem than Josquin
CONFIRMED: Less music for more than 4 voices
CONFIRMED: More 3-voice music
CONFIRMED: More triple meter
SAME: Less stepwise motion
SAME: More notes at the bottom of the range
SAME: More chords (or simultaneities) without a third
SAME: More varied rhythmic note values
OPPOSITE: More large leaps (larger than a 5th)
OPPOSITE: More dissonance

Figure 4. Results of empirical testing of expert predictions. "CONFIRMED" means the expectations were empirically correct, "SAME" indicates no statistically significant difference between the two composers and "OPPOSITE" means the expected characteristic was infact more associated with Josquin than Ockeghem.

Next, in order to demonstrate the types of novel musicological insights jSymbolic's features can reveal, Weka was used to apply seven statistical feature analysis techniques (based on feature-class correlations, information gain, etc.) to highlight the features that most effectively distinguish the composers in each of the two composer pairs. The results were compiled into ranked feature lists.

It turns out that a combination of rhythmic characteristics are particularly important in distinguishing Josquin from Ockeghem and, furthermore, Ockeghem tends to have more vertical sixths and diminished triads, as well as longer melodic arcs. With respect to Josquin and La Rue, Josquin tends to have: more vertical unisons and thirds; fewer vertical fourths and octaves; and more melodic octaves.

6. CONCLUSIONS

jSymbolic is a powerful and accessible tool that music researchers can apply to diverse research areas and types of music. It can also serve as a platform that researchers can use to develop their own bespoke features. It is hoped that this will help address the paucity of symbolic music software produced by the MIR community to date, relative to the extensive range of software it has produced associated with audio and other data, and will encourage greater MIR engagement with musicologists and music theorists. jSymbolic's easy-to-use interfaces and extensive documentation are intended to facilitate this.

Although jSymbolic features can certainly be used in classification tasks, as in the experiments described above, the direct study of feature values also has important potential. Such work can combine expert manual study with the use of statistical analysis techniques. Research can consist of empirical validation of existing hypotheses or of purely exploratory research, all involving the study of potentially huge quantities of music. Both approaches can help scholars arrive at initially unintuitive but potentially crucial musicological insights.

In terms of experimental conclusions, the results from Sections 4 and 5 permit the following observations:

- The new jSymbolic 2.2 features were quite effective in both genre and composer classification, even using generic machine learning approaches. They were able to distinguish between seven Renaissance composers 92.4% of the time, and achieved 93.2% genre classification accuracy when applied to a 5-genre ontology, and 77.6% when classifying amongst 10 genres.
- The new jSymbolic 2.2 features produced better classification accuracies than the old jSymbolic 1.2 features in all tests, and better F-scores in all but one.
- The new jSymbolic 2.2 features were effective in testing expert expectations about differences in the musical styles of pairs of Renaissance composers, and in revealing additional unanticipated differences.

7. FUTURE WORK

We will continue to work with musicologists and music theorists by helping them carry out research on large musical datasets with jSymbolic. We will also assist them in implementing specialized features of their own. A particular focus of this collaborative work will be placed on the determination of which features are most effective in distinguishing different musical classes (composers, genres, regions, etc.), and on investigating why. We will also expand our work on using machine learning to help resolve controversial composer attribution. We also intend to work on expanding the extent to which jSymbolic can extract features from non-Western musics by adding still more relevant features.

We are currently working on integrating jSymbolic2 into the SIMSSA/MIRAI architecture [10], so that researchers can search the project's rich music databases using content-based queries formulated using feature values and ranges. A researcher could thus filter results based on the amount of chromaticism in a piece, for example, or the amount of parallel motion between voices. Of even greater interest, queries could potentially be formulated based on hard-to-quantify high-level characteristics, such as degree of tonality, made possible by machine learning models trained on jSymbolic features.

Related to this project, we also plan to apply jSymbolic to symbolic files that have been generated using optical music recognition or automatic audio transcription software, and to investigate the robustness of the features to such error-prone data. In addition to making an enormous amount of new music available for symbolic feature extraction, doing this successfully would also greatly facilitate multimodal research. The huge Lakh dataset [18] can also be studied with similar goals.

An additional priority will be to add new parsers so that features can be extracted from additional file formats. We are especially looking at adding features specially designed for music encoded using mensural or other early music notations. Finally, we intend to work towards porting jSymbolic2's new features to other platforms, especially music21 [4].

8. ACKNOWLEDGEMENTS

We would like to thank the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Fonds de recherche du Québec - Société et culture (FRQSC) for their generous funding. We would also like to acknowledge the formidable contributions of our many collaborators on the MIRAI and SIMSSA projects, especially Tristano Tenaglia.

9. REFERENCES

[1] A. Brinkman, D. Shanahan and C. Sapp, "Musical stylometry, machine learning and attribution studies: A semi-supervised approach to the works of Josquin," *Proc. of the Biennial Int. Conf. on Music Perception and Cognition*, pp. 91–97, 2016.

[2] D. C. Corrêa and F. A. Rodrigues, "A survey on symbolic data-based music genre classification," *Expert Systems with Applications*, Vol. 60, pp. 190–210, 2016.

[3] J. E. Cumming et al., "Methodologies for creating symbolic corpora of Western music before 1600," *Proc. of the Int. Soc. for Music Information Retrieval Conf.*, accepted for publication, 2018.

[4] M. S. Cuthbert, C. Ariza and L. Friedland, "Feature extraction and machine learning on symbolic music using the music21 toolkit," *Proc. of the Int. Soc. for Music Information Retrieval Conf.*, pp. 387–92, 2011.

[5] J. S. Downie et al., "MIREX2017 Results - MIREX Wiki," Music-ir.org, 2016. [Online]. Available: http://www.music-ir.org/mirex/wiki/2017:MIREX2017_Results. [Accessed: 28-March-2018].

[6] T. Eerola and P. Toiviainen, "MIR in Matlab: The MIDI Toolbox," *Proc. of the Int. Conf. on Music Information Retrieval*, pp. 22–7, 2004.

[7] A. Hankinson, "Optical music recognition infrastructure for large-scale music document analysis," Ph.D. diss., Schulich School of Music, McGill Univ., Montreal, Canada, 2015.

[8] D. Huron, "Music information processing using the Humdrum toolkit: Concepts, examples, and lessons," *Computer Music J.*, Vol. 26, No. 2, pp. 11–26, 2002.

[9] C. McKay, "Automatic music classification with jMIR," Ph.D. diss., Schulich School of Music, McGill Univ., Montreal, Canada, 2010.

[10] C. McKay and I. Fujinaga, "Building an infrastructure for a 21st-century global music library," *Int. Soc. for Music Information Retrieval Conf. Late Breaking and Demo Papers*, 2015.

[11] C. McKay et al., "Using statistical feature extraction to distinguish the styles of different composers," *45th Medieval and Renaissance Music Conf.*, 2017.

[12] C. McKay et al., "Characterizing composers using jSymbolic2 features", *Extended Abstracts for the Late-Breaking Demo Session of the Int. Soc. for Music Information Retrieval Conf.*, 2017.

[13] C. McKay, "jMIR," SourceForge.net, SourceForge.net, 2018. [Online]. Available: http://jmir.sourceforge.net. [Accessed: 6-June-2018].

[14] C. McKay, "jSymbolic Manual," SourceForge.net, 2018. [Online]. Available: http://jmir.sourceforge.net/manuals/jSymbolic_manual/home.html. [Accessed: 6-June-2018].

[15] P. J. Ponce de León and J. M. Iñesta, "Statistical description models for melody analysis and characterization," *Proc. of the Int. Computer Music Conf.*, pp. 149–56, 2004.

[16] P. Roland, "The music encoding initiative (MEI)," *Proc. of the First Int. Conf. on Musical Applications Using XML*, pp. 55–9, 2002.

[17] C. Raffel, and D. P. W. Ellis, "Intuitive analysis, creation and manipulation of MIDI data with pretty_midi," *Int. Soc. for Music Information Retrieval Conf. Late Breaking and Demo Papers*, 2014.

[18] C. Raffel. "Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching," Ph.D. diss., Columbia Univ., New York, USA, 2016.

[19] J. Rodin, C. S. Sapp and C. Bokulich, "The Josquin Research Project," Stanford Univ., 2017. [Online]. Available: http://josquin.stanford.edu. [Accessed: 28-March-2018].

[20] A. Sigler, J. Wild and E. Handelman, "Schematizing the treatment of dissonance in 16th-century counterpoint," *Proc. of the Int. Soc. for Music Information Retrieval Conference*, pp. 645–51, 2015.

[21] B. L. Sturm, "A simple method to determine if a music information retrieval system is a 'horse'," *IEEE Trans. on Multimedia*, Vol. 16, No. 6, pp. 1636–44, 2014.

[22] D. Temperley, *The cognition of basic musical structures*. Cambridge, MA: MIT Press, 2001.

[23] E. Verdaguer Morales, "Anàlisi i generació algorísmica de línies de baix en estil Funk," Thesis, Pompeu Fabra Univ., Barcelona, Spain, 2014.

[24] I. H. Witten, E. Frank and M. A. Hall, *Data mining: Practical machine learning tools and techniques*. New York: Morgan Kaufman, 2011.