CHORD RECOGNITION IN SYMBOLIC MUSIC:

A SEGMENTAL CRF MODEL, SEGMENT-LEVEL FEATURES,
AND COMPARATIVE EVALUATIONS ON CLASSICAL AND POPULAR MUSIC

____________________________________

A Thesis
Presented to
The Honors Tutorial College
Ohio University

____________________________________

In Partial Fulfillment
of the Requirements for Graduation
from the Honors Tutorial College
with the Degree of
Bachelor of Science in Computer Science

____________________________________

by
Kristen S. Masada
April 2018
Acknowledgements

In addition to the acknowledgements mentioned in our attached journal manuscript, I would like to thank my HTC thesis adviser, Dr. Razvan Bunescu. Before our first tutorial in fall of 2016, I knew that I enjoyed computer science, but I felt lost in terms of what I wanted to pursue after graduation. I was also frequently shy and often avoided speaking to my professors, even in tutorials. I am unbelievably grateful that I had the chance to have four tutorials with him, take three classes with him (including machine learning and deep learning), work under him for my senior design project, publish an international conference paper with him, and co-author a journal submission with him within the past two years. His passion for both machine learning and music, as well as his dedication to research, inspired me to pursue and achieve things that I otherwise thought I would not be capable of accomplishing as an undergraduate. I am very excited to be able to continue doing music information retrieval research with him as a Master’s student next year.
Table of Contents

1. Introduction..............................................................................................................4
2. Music Theory Background......................................................................................6
3. Extended Literature Review of Existing Chord Recognition Systems................10
4. Machine Learning Background.............................................................................15
    4.1. Semi-Markov Conditional Random Fields....................................................16
5. Extended Methodology...........................................................................................19
6. References...............................................................................................................23
7. Co-Author Contributions..........................................................................................25
8. Proof of Journal Submission...................................................................................26
9. Journal Submission.................................................................................................28
1. Introduction

Music analysis is a field of study that consists of annotating specific elements of a piece of music and determining how these elements contribute to the overall meaning of a song. Chord recognition—the identification of groups of notes “that make sense when played or sung all at the same time”—represents an important subtask of music analysis, as this information is critical in determining the overall harmonic structure of a song [1]. In recent decades, the use of computer science to aid music analysis has led to the emergence of a variety of automatic chord recognition systems for symbolic music (i.e. written sheet music). The earliest system from the 1960’s frames automatic music analysis as a purely academic problem [13]. Conversely, more recent systems have implications for popular music streaming services such as Spotify, in which chord recognition is useful in performing complex tasks such as music identification, music similarity computation, and automatic playlist generation [10]. These newer systems are also tailored to the field of music information retrieval, with applications to more advanced research problems such as automatic accompaniment and automatic composition [9] [10]. Finally, they are useful for music annotators, professors, and students, who spend a great deal of time manually annotating chords for educational purposes [10].

The most advanced automatic chord recognition system proposed for symbolic music in recent past is Radicioni and Esposito’s HMPerceptron system from 2010. This system effectively demonstrates the convenience and flexibility of a machine learning-based system over prior systems, achieving an 80.06% accuracy rating on the Bach Choral Harmony Dataset (BaCh)—a set of 60 Bach chorales [10]. However, it possesses one major shortcoming: it does not directly address the problem of segmentation—the
identification of segments of a musical piece that correspond to a particular chord—that Bryan Pardo and William Birmingham do with their HarmAn algorithm from 2002 [9]. Thus, in this thesis, I propose a new automatic chord recognition system called semi-CRF that incorporates both of these elements: a machine learning-based system that directly models segmentation. Through extensive analysis on BaCh and three other datasets, I demonstrate that semi-CRF consistently outperforms HMPerceptron on both event and segment-level chord accuracy. I also analyze both systems, as well as an additional system called the Melisma Music Analyzer proposed by David Temperley, on event and segment-level chord root accuracy. Semi-CRF outperforms HMPerceptron for all datasets on this task, and outperforms Melisma for three out of four datasets.

Rather than writing a traditional thesis, I have chosen to write a journal submission. This submission is titled “Chord Recognition in Symbolic Music: A Segmental CRF Model, Segment-Level Features, and Comparative Evaluations on Classical and Popular Music” and is co-authored with my thesis adviser, Dr. Razvan Bunescu¹. I have chosen to submit our work to the Transactions of the International Society for Music Information Retrieval (TISMIR)—a journal that was recently established by the International Society of Music Information Retrieval Conference (ISMIR). I made this decision based on the fact that ISMIR represents the top conference for music information retrieval, and I am confident that the level of quality of their conference work will transfer to their new journal. Dr. Bunescu and I also already published a conference paper in ISMIR on our automatic chord recognition system in October of 2017 [3]. Thus, we believe that we have a good chance of having this paper accepted, as it is an extended version of this conference paper.

¹ See Section 7 (‘Co-Author Contributions’) for details of each author’s contributions to the journal submission.
As the attached journal manuscript contains the core of the information about our semi-CRF system, this document will act as a supplement to the submission in two forms. First, it will provide descriptions of music theory (Section 2) and machine learning (Section 4) for a lay audience, offering a friendlier introduction to these topics than what is provided in the journal paper. Second, it will contain an extended version of the literature survey (Section 3) and methodology (Section 5) provided in our submission. The former will discuss previous systems mentioned in the paper in more detail, as well as descriptions of some systems that are not mentioned. The latter will contain implementation details of our system that I was not able to include in the submission due to TISMIR’s word limit.

2. Music Theory Background

![Image](image.png)

*Figure 1: A C major chord (left) and a C major 7 chord (right).*

In music, a **chord** is a group of three or more notes “that make sense when played or sung all at the same time” [1]. The prototypical instance of a chord is the triad, in which a third and a fifth are created on top of the chord’s lowest note—the **root** [2]. Figure 1 includes an example of a C chord on the left, in which the lowest note (C) is the root, the note above this (E) is the third, and the highest note (G) is the fifth. Each chord possesses a **mode** as well. In general, the mode is determined by the **interval**—the number of half steps or black and white keys on a piano—between the root and the third. The interval between the root and the fifth is also useful in determining the mode, but is generally less
critical. The C chord previously discussed in Figure 1 is a major chord, as the interval between C and E forms a major third (4 half spaces), while the interval between C and G forms a perfect fifth (7 half spaces). While triads are most common, it is also possible for chords to contain an added note, or a fourth note. For instance, the chord on the right side of Figure 1 contains a C major 7 chord: a C major chord augmented with a B, which forms a major seventh (11 half steps) with C.

![Figure 2: (a) A C major chord in first inversion. (b) A C major 7 chord with a missing fifth. (c) Two C major chords with a passing tone (F) in between. (d) An arpeggiated C major chord. (e) An example of a group of notes that form either an F major chord](image)

While the previous discussion may offer the impression that chord recognition is as simple as identifying the individual notes within a given segment and determining the corresponding chord label, the task is complicated by a variety of factors. It is possible for a chord to be inverted so that the root does not occur as the lowest note. Figure 2a. shows an example of a C major chord in first inversion, in which the E appears as the lowest note instead of the C. Chords may also be missing notes when they appear in a song. For instance, the chord shown in Figure 2b. is still a C major 7 chord despite the fact that it is missing its fifth (G), as this interval is less defining to the overall sound of a seventh chord in comparison with a third or seventh. Conversely, non-harmonic tones—additional notes that do not belong to any chord—may appear within a piece. These notes may constitute one of several kinds of common non-harmonic tones, such as passing tones and neighboring tones, but are generally still difficult to distinguish from added notes that
form part of a chord. Figure 2c. includes an example of two C major chords with a note (F) that operates as a passing tone between the E of the first chord and the G of the second chord. Arpeggios, in which one note of the chord occurs after the other instead of simultaneously, are another factor that makes the identification process more challenging. Figure 2d demonstrates an arpeggiated C major chord. Finally, many chords share notes, making it difficult to determine which of these chord labels is right. For example, the notes in Figure 2e form both an F major chord in first inversion and an A minor 6 chord that is missing a fifth (E).

Figure 3: A segment that contains the notes of both a C major 7 chord and an E minor 6 chord.

The sharing of notes between chords begs an important question: how does a musician know which label is correct when the notes that appear in a section of music correspond to more than one chord? In such cases, it is useful to consider the duration of each note, as notes that are sustained for longer periods of time are typically more prominent to the listener than those that are short in duration [5]. Figure 3 shows a segment that contains a group of notes that could either form a C major 7 chord or an E minor 6 chord. In this example, C and E both appear as whole notes (i.e. open circles), which each have a duration of four beats in the 4/4 time signature shown. Conversely, G and B form quarter notes (i.e. closed circles with protruding vertical lines), which last for one beat each with respect to the same time signature. As the duration of C and E are four times as long as the other notes within the segment, it is likely that these notes represent the root and third of the correct chord label, rather than less critical notes of the chord, such as the
fifth or added note. Thus, it is more likely that the correct label for this segment is C major 7, rather than E minor 6, in which E forms the root, but C represents the added sixth. The accent of each note in a segment—the perceptual prominence of the beat it occurs on—can be used similarly [5]. In the previous example, C and E sound on the first, or strongest, beat of the measure and are thus more likely to stand out to the listener. In contrast, G appears on the third beat, which is slightly weaker than the first. Finally, B appears on the fourth beat: the weakest beat. This again points to the fact that C and E are likely to be the root and third of the underlying chord.

![Figure 4: A C major scale, which consists of the notes C, D, E, F, G, A, B, and C.](image)

**Figure 4:** A C major scale, which consists of the notes C, D, E, F, G, A, B, and C.

![Figure 5: The diatonic chords of the key of C major: C major, D minor, E minor, F major, G major, A minor, and B diminished.](image)

**Figure 5:** The diatonic chords of the key of C major: C major, D minor, E minor, F major, G major, A minor, and B diminished.

Besides duration and accent information, the overall key of a work and the chord label of the previous segment in the piece are helpful in determining the correct chord in cases where the notes of a segment correspond to more than one chord. A musical key is a group of pitches that correspond to a scale. In turn, a scale is a group of ascending pitches that spans the interval of an octave (12 half spaces) [5]. Figure 4. shows a C major scale, which contains the notes C, D, E, F, G, A, and B in ascending order. These notes also form the key of C major. Using the notes within this key, it is possible to build a triad on top of
each scale degree, as shown in Figure 5. These triads represent the diatonic chords of the key of C major—chords built exclusively from the scale tones of the given key. Among these chords, some chord transitions are more common than others, which can be useful in determining the right label of a segment [5]. For example, if a C major triad appears in the previous segment of a piece written in the key of C major, it is more likely that the next segment is a G major chord—the root note of which forming the fifth degree of the C major scale—rather than an A minor triad—the root forming the sixth scale degree, a less stable tone than the fifth.

When the key of a musical work is known, each chord label can be converted to Roman numeral notation, so that each label is replaced with a number representing the scale degree of the root of the chord in the given key. For instance, a G major chord appearing in a piece written in the key of C becomes a V chord, since G represents the fifth degree of the C major scale. This Roman numeral notation represents a higher level harmonic analysis of a piece, as the harmonic function of a chord is now contained within the label. This information is useful in a variety of applications, such as analyzing a composer’s style or determining the genre of a piece of music based on the chord progressions that appear in the piece [10].

3. Extended Literature Review of Existing Chord Recognition Systems

The field of computerized harmonic analysis has progressed dramatically over the past several decades, not only in the technical complexity of the algorithms introduced, but also in the way in which researchers have conceptualized the task of harmonic analysis. A
survey of five significant works within the field effectively demonstrates the overall shift from the usage of natural language processing analytical methods to music theory-heavy, algorithmically sophisticated systems: Terry Winograd’s 1968 grammar-based; Stephen Smoliar’s 1980 LISP program that performs a unique form of harmonic analysis called Schenkerian analysis; H. J. Maxwell’s detail-heavy, rule-based system for automatic chord recognition from 1992; Bryan Pardo and William Birmingham’s 2002 algorithms that represent a generic platform for performing automatic chord recognition; and Daniele Radicioni and Roberto Esposito’s 2010 machine learning-based chord recognition system. Of the five, Radicioni and Esposito’s system represents the most convincing and promising of approaches to performing the task of automatic harmonic analysis, as it successfully demonstrates the convenience and adaptability of machine learning-based approaches over prior systems.

The earliest significant work to describe a computer program that performs harmonic analysis is Terry Winograd’s 1968 paper titled “Linguistics and the Computer Analysis of Tonal Harmony” [13]. In this, Winograd details a LISP program that identifies the chords of a piece of music in Roman numeral notation—a type of notation that relates each chord to the overall key of the piece. As the title of his paper implies, the main purpose of Winograd’s research is to obtain a better understanding of both “the structure of music and the use of semantic techniques in parsing arbitrary languages” [13, 42]. Thus, Winograd mainly employs analytical methods typically characteristic of natural language processing in analyzing pieces of music, utilizing a newer form of grammar—systemic grammar—to represent tonal harmony as a language with its own structural rules for his LISP program to parse [13, 27]. He also mostly references linguists in his bibliography,
the most notable of which being Noam Chomsky, co-founder of the generative grammar theory, and Allen Forte, one of the first researchers of systemic grammars. In general, it is understandable for a programmer such as Winograd to view automated music analysis from a linguistic perspective during this time period, as a plethora of artificial intelligence research existed in the area of natural language processing at this time, while research in computerized music information retrieval was nearly nonexistent.

Stephen Smoliar’s “A Computer Aid for Schenkerian Analysis” from 1980 is the next major work to appear in the field of computerized harmonic analysis [12]. Smoliar explains at the beginning of his paper that his LISP program performs an entirely different kind of harmonic analysis than Winograd’s: chord significance, which involves using Schenkerian techniques to determine the “special, architectonic purpose of a chord within a phrase” [12, 41]. He notes that Winograd’s program merely performs chord grammar analysis, labeling each chord within a phrase, but not explaining its overall function within the piece aside from its relation to the key. Similar to Winograd though, Smoliar borrows analytical methods from natural language processing, as there is still a dearth of artificial intelligence research involving music during this time period. Most notably, he references Chomsky’s theory of transformational grammars to describe how his program can transform a sequence of chord tones into a more complex phrase that contains ornamental notes. Smoliar again differs from Winograd in his conclusion, stating that the “significance of Schenkerian analysis is, and should always be, within the rightful domain of music theory,” thus demonstrating that the overall purpose of his work is to advance the field of music, and not linguistics [12, 48]. In general, Smoliar’s work does not build on the technical complexity of Winograd’s program, as it involves repeatedly applying simple
commands to a sequence of notes in LISP to produce the overall Schenkerian analysis of a piece. However, it does demonstrate a successful implementation of an algorithm that performs a different kind of harmonic analysis besides typical chord grammar analysis.

H. J. Maxwell’s 1992 “An Expert System for Harmony Analysis of Tonal Music” builds upon Winograd’s work in detailing a complex LISP program that performs chord grammar analysis using Roman numeral notation [4]. Overall, Maxwell’s system is more autonomous than Winograd’s. For example, Winograd provides the key of a piece as part of the input to his program, as well as the vertical groups of notes to be labeled with a chord. In contrast, Maxwell’s system determines regions of tonality and chord verticalities—groups of notes sounding at each note onset or offset—within a piece of sheet music on its own. Maxwell also implements a wider breadth of music theory in his program, utilizing a set of fifty-five rules to identify the verticalities and their respective chord labels. The most notable aspect of these rules is their consideration of meter and accentuation, which Winograd’s system entirely ignores. Thus, Maxwell’s system is able to consider musical subtleties more advanced than prior systems, allowing for a more complex analysis of the chords within a piece of music. Maxwell also states that the overall purpose of his research is to attempt to build a program that mimics the thought process of a professional music analyst. This explains why he does not take inspiration from linguists in the same way that Winograd and Smoliar do. His goal is also reflective of the fact that computerized music information retrieval is an established field within artificial intelligence at this point, and therefore is not as dependent on the area of natural language processing as it was before.

Bryan Pardo and William P. Birmingham introduce a notable automatic chord recognition system to the field of computerized harmonic analysis in their 2002 paper
“Algorithms for Chordal Analysis” [9]. Pardo and Birmingham explain that the purpose of their paper is to provide a general algorithm for performing harmonic analysis for others to build upon. Thus, their system does not incorporate music theory concepts as complex as Maxwell’s system, avoiding metrical analysis and ignoring the key of the piece. What is most notable about Pardo and Birmingham’s work is the fact that they identify chord labeling and segmentation—“splitting the music into appropriate chunks (segments) for analysis”—as separate subproblems within the task of computerized harmonic analysis [9, 27]. As previously stated, Winograd does not consider segmentation as an issue in that he provides the verticalities of the piece of music he is analyzing as input to his system. Maxwell is more considerate of this in the sense that his program is able to find individual vertical events to be labeled within a piece, although it is incapable of identifying adjacent groups of events that are harmonically similar prior to the labeling process. Pardo and Birmingham also employ more sophisticated algorithmic techniques than their predecessors in implementing their system, designing their own HarmAn algorithm which uses a “greedy” approach to consider as few segmentations as possible in determining the optimal segmentation of a piece of music. In general, the fact that Pardo and Birmingham purposefully construct a general system for others to specialize is demonstrative of the fact that automatic harmonic analysis is no longer a niche area of research at this point: it has developed into an established area within music information retrieval with a wide variety of applications, such as automatic composition and computerized genre classification.

Finally, Daniele P. Radicioni and Roberto Esposito describe an automatic harmonic analysis system in their 2010 paper “BREVE: An HMPerceptron-Based Chord Recognition System” that performs the same kind of chord grammar analysis as its
predecessors, but utilizes the most complex programmatic implementation in the field to date: machine learning [10]. This approach differs from a typical computer science algorithm by allowing the computer to learn how to perform chord recognition on its own. Thus, the computer can figure out idiosyncrasies involved in the task that would otherwise be difficult for a programmer to specify in a set of instructions. For instance, it can easily calculate weights to determine how frequently chord changes occur on the first beat of a measure and how often verticalities with a bass note as the third degree of the chord label in consideration are identified as chords in first inversion. This convenience mitigates the necessity for writing large amounts of code to implement complex, interrelated rules in determining chord labels—such as what Maxwell does—while still allowing the system to take large amounts of characteristics about the data into consideration. It is important to mention that Radicioni and Esposito’s system is event-based like Winograd’s system, taking pre-defined verticalities as input for labeling. In this respect, it is less advanced than Maxwell’s verticality detection system and Birmingham and Pardo’s segmentation algorithm. It is also significant to note that Radicioni and Esposito are not split between two fields in the way that Winograd is in the purpose of his research; rather, they are most interested in advancing the field of music information retrieval—machine learning merely serves as a tool in this process. Again, this is indicative of the fact that automatic harmonic analysis is substantial enough of a research area to stand on its own.

4. Machine Learning Background

According to Andrew Ng—the founder of the Google Brain Deep Learning Project—machine learning is the “science of getting computers to act without being
explicitly programmed” [7]. There are multiples kinds of machine learning, of which \textit{supervised learning} is most relevant to the learning model I used for my thesis. This type of learning consists of providing a collection of \textit{training examples} to the algorithm that also includes the correct label for each of these examples. In the case of chord recognition, a training example is a song annotated with the correct chord labels. The learning algorithm then calculates a \textit{hypothesis}, or function, based on these labels, which it can use to predict the labels for \textit{test cases} that it has not seen before. The calculated hypothesis is a function of a set of \textit{features}: a collection of characteristics that describe the data being analyzed. An example of a useful feature for the task of automatic chord recognition is root coverage: a feature that indicates if the root note of the candidate chord label appears in a given segment of a piece of music. In general, features are crucial to the performance of a machine learning system, as features that do not represent the most prominent characteristics of the input data accurately enough will lead to poor performance on unseen data [8].

\section*{4.1. Semi-Markov Conditional Random Fields (semi-CRF)}

For my thesis, I use a \textit{semi-Markov Conditional Random Fields (CRF)} learning model to represent the problem of chord recognition. As explained in Sunita Sarawagi and William W. Cohen’s 2004 paper “Semi-Markov Conditional Random Fields for Information Extraction,” a semi-Markov CRF is capable of splitting an input sequence into segments and labeling these segments such that it takes into account correlations between the labels of consecutive segments. Thus, by using this learning algorithm, I am able to identify segments of music that correspond to a given chord label, rather than using an
event-based implementation like BREVE’s HMPerceptron model. The equation below shows the probability distribution that a semi-Markov CRF defines:

\[ P(s, y|x, w) = \frac{e^{w^T F(s, y, x)}}{Z(x)} \]  (1)

In this formula, \( s = (s_1, s_2, ..., s_K) \), or the sequence of segments that make up an input \( x \)—in this case, a song. A single segment \( s_n \) within \( s \) is defined as \( (s_n.f, s_n.l) \), in which \( s_n.f \) is the first event of a segment and \( s_n.l \) is the last event. For chord recognition, I define a musical event—or chord verticality—as the notes sounding at every unique note onset or offset in a piece of music. Figure 6 demonstrates the difference between an event and a segment in a piece of music. Additionally, \( y = (y_1, y_2, ..., y_K) \), or the sequence of correct chord labels that correspond to the segmentation \( s \) of song \( x \), and \( w \) is a vector of parameters, one for each feature in \( F(s, y, x) \). Therefore, \( P(s, y|x, w) \) is the probability that \( s \) and \( y \) represent the right segmentation and set of chord labels for song \( x \), respectively [11].

\( F(s, y, x) \) in equation (1) can be expanded to the following formula:

\[ F(s, y, x) = \sum_{k=1}^{K} f(s_k, y_k, y_{k-1}, x) \]  (2)

Thus, \( F(s, y, x) \) is the sum of local segment feature vectors \( f(s_k, y_k, y_{k-1}, x) \), in which \( y_k \) is the label for the current segment \( s_k \) and \( y_{k-1} \) is the label of the previous segment. The normalization constant \( Z(x) \) in equation (1) can also be expanded:

\[ Z(x) = \sum_{s,y} e^{w^T F(s, y, x)} \]  (3)

\( Z(x) \) represents a summation computed over all possible segmentations \( s \) and labelings \( y \) of input \( x \), in which each global feature vector \( F(s, y, x) \) is multiplied with parameters \( w \) and is used as an exponent for constant \( e \). To compute the probability \( P(s, y|x, w) \) for a
specific segmentation and configuration of labels, this also requires raising the constant \( e \) to the power of \( w^TF(s, y, x) \), but only for the global feature vector specific to this segmentation \( s \) and label configuration \( y \). This overall value is then divided by normalization constant \( Z(x) \). At test time, the goal of using this semi-Markov CRF model is to maximize \( P(s, y|x, w) \) to determine both the correct segmentation and sequence of labels for an input \( x \) [11]. A summary of the variables included in equations 1 and 2 is shown in Table 1 below.

<table>
<thead>
<tr>
<th>( x )</th>
<th>An input; in this case, a song.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s )</td>
<td>A segmentation of song ( x ), represented as the vector of segments ( \langle s_1, s_2, \ldots, s_K \rangle ), in which ( K ) represents the total number of segments.</td>
</tr>
<tr>
<td>( y )</td>
<td>A sequence of chord labels that correspond to the segmentation ( s ) of song ( x ).</td>
</tr>
<tr>
<td>( w )</td>
<td>A vector of calculated weights that correspond to each feature value of ( x ).</td>
</tr>
<tr>
<td>( Z(x) )</td>
<td>Normalization constant.</td>
</tr>
<tr>
<td>( P(s, y</td>
<td>x, w) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( s_k )</th>
<th>The current segment.</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_k )</td>
<td>The label of the current segment.</td>
</tr>
<tr>
<td>( y_{k-1} )</td>
<td>The label of the previous segment.</td>
</tr>
<tr>
<td>( f(s_k, y_k, y_{k-1}, x) )</td>
<td>Local segment feature vector.</td>
</tr>
<tr>
<td>( F(s, y, x) )</td>
<td>Global segment feature vector—the sum of local segment feature vectors.</td>
</tr>
</tbody>
</table>

Table 1: A summary of each variable in equations 1, 2, and 3.

To implement my chord recognition system, I used a semi-Markov CRF package written in Java that has previously been used for a noun phrase chunking task [6]. My thesis adviser, Dr. Razvan Bunescu, primarily chose this package because it simplifies the
definition of semi-Markov CRFs provided in equations (1) and (2) to perform training and testing more quickly. This new definition restricts local segment features to two types: segment-label features \( f(s_k, y_k, x) \) that rely on the current label and segment boundaries in consideration and transition features \( g(y_k, y_{k-1}, x) \) that rely on the labels of the current and previous segments. These new formulas are provided below:

\[
P(s, y|x, w) = \frac{e^{w^T F(s, y, x) + u^T G(s, y, x)}}{Z(x)} \quad (3)
\]

\[
F(s, y, x) = \sum_{k=1}^{K} f(s_k, y_k, x) \quad (4)
\]

\[
G(s, y, x) = \sum_{k=1}^{K} g(y_k, y_{k-1}, x) \quad (5)
\]

Note that a separate weight vector is trained for each type of feature: \( w \) is calculated for global segment-label feature vector \( F(s_k, y_k, x) \) and \( u \) is calculated for global transition feature vector \( G(y_k, y_{k-1}, x) \).

5. Extended Methodology

![Figure 7: An example of the kind of candidate segments semi-CRF computes. y indicates the chord label, s.f is the index of the first event in the segment and s.l is the index of the last event in the segment.](image)

Many implementation considerations went into our system that were not explicitly mentioned in our journal manuscript, or were not described in full detail. A primary example of this is the issue of maximum segment length. When finding the most likely boundaries for each segment in a given example, semi-CRF computes a candidate segment
with length one, two, up to this maximum segment length starting from each event in the song. A simple example of this is included in Figure 7. If the set maximum segment length is shorter than the length of the actual maximum segment in the entire dataset, then it will not be able to produce a single label and segmentation for this segment. However, on the flip side of this, setting a high maximum segment length greatly increases the number of nodes in the probabilistic graph that semi-CRF generates, where each node is a candidate segment. This significantly increases the amount of memory that the system requires, as I discovered when setting the maximum segment length to the true maximum length in the TAVERN dataset (around 250 events). As a compromise between maintaining a relatively long maximum segment length while still keeping memory usage at bay, Dr. Bunescu and I decided to break long segments into chunks of 20 events. When evaluating the segment-level accuracy at test time, we combine any consecutive segments with the same chord label to produce a single segment with the lengths of these constituent segments combined. In addition to TAVERN, we discovered that both KP Corpus and the Rock dataset contained a number of segments of length greater than 20 events. Thus, we ended up using this evaluation method for these datasets as well.

Figure 8: An example of a long segment, where the notes between the first two dashed lines form a segment of length 20 and the notes between the second pair of dashed lines form a short residual segment. Despite the fact that the label is C:maj7 for the whole segment, the residual segment does not contain the root note C, thus creating a segment where the root is missing.
We experienced another issue involving maximum segment length when evaluating the Rock dataset. As we discuss in Section 5.4 of our journal manuscript, we found that the original labels in the rock dataset contained some errors. For instance, there were segments that did not contain the root of the chord label specified, and other segments that did not contain the added note included in its label. I ran our system to automatically detect these sorts of issues throughout the dataset. However, I noticed that some of the segments that were supposedly missing a root note were in fact very short residual segments occurring at the end of a longer segment that had been divided into chunks of 20 events. An example of this is shown in Figure 8 on the previous page. We initially thought to add the length of the last chunk of 20 events to the length of the residual block and divide this in half to mitigate the problem. However, we realized that this would not work—providing semi-CRF with consecutive segments of the same label with varying segment lengths would give it an arbitrary sense of when segments should start and end. We then produced a sort of “hack” to maintain a fixed length for breaking longer segments: we tried possible values for the maximum segment length between 10 and 30 and calculated the number of residual segments that this would produce of length less than 5 on the entire Rock dataset. We then chose the value that produced the most reasonable trade-off between memory usage and a limited number of short residual segments. In the end, we used 26 as our maximum segment length for evaluating the Rock dataset, as it produced only 16 segments with a length less than 5.

A more minor implementation detail omitted from the paper is our experimentation with different hyperparameter settings for the full chord evaluation on the KP Corpus for semi-CRF. We briefly mention in Section 6.1 of the journal manuscript that we only
include a feature in our system if it occurs five times or more in the gold segments of the training data. However, for the KP Corpus, we also tried increasing this minimum threshold count to ten and fifteen. We discovered that increasing this threshold to ten increased the event-level accuracy of semi-CRF by nearly 2% over the original version of the system that used a threshold of five. Thus, we ended up using this setting in the results for both versions of semi-CRF that appear in Table 5 in the journal paper. We additionally tried tuning the L2 regularization parameter, increasing it from 0.125 to 0.250 and 1.250. However, each setting either produced the same or worse results. Lastly, instead of entirely including or omitting the augmented 6th chord results, we experimented with different levels of including augmented 6th chord features. For example, we included only the unweighted augmented 6th chord features in one experiment, while we included only the augmented 6th coverage features in another. Similar to before, these modifications did not improve the overall accuracy.
6. References


7. Co-Author Contributions

In terms of the implementation of semi-CRF, I interfaced Muis and Lu’s semi-CRF package to be able to perform chord recognition entirely on my own. This amounted to writing over 9,400 lines of code in 13 Java files. I also modified HMPerceptron’s Objective-C code slightly to be able to read in training and testing folds. Additionally, I wrote a significant number of Python scripts for various tasks. One of these tasks was to convert MusicXML files to the input format of our system, as well as that of HMPerceptron and Melisma. Another was to correct the chord labels of the KP Corpus dataset, which was missing some labels and had many labels with the incorrect onset time. Yet another was to analyze the accuracy of the output of each system for both chord and chord root evaluation. A more minor one was to generate the training and testing fold files for each dataset for semi-CRF. The only part of the implementation that I did not do myself was converting the TAVERN dataset from a format called standard **kern to MusicXML. Former Master’s student Patrick Gray did this for me.

In terms of the journal paper, Dr. Bunescu and I each wrote roughly 50% of the paper. It is difficult to come up with an exact number, as we collaborated on the ideas of most sections and edited each other’s writing. Specifically though, I wrote Section 2 (‘Types of Chords in Tonal Music’), half of Section 4 (‘Chord Recognition Features’), the last half of Section 5 (‘Chord Recognition Datasets’), and Section 6 (‘Experimental Evaluation’) minus some of the error analysis, which Dr. Bunescu and I collaborated on. Dr. Bunescu wrote the rest of the sections in the paper and also created the figures. Ultimately, the idea to apply semi-CRFs to the task of symbolic chord recognition was Dr. Bunescu’s. However, we jointly collaborated on the design of the features for our system.
8. Proof of Journal Submission

Proof of submission of our journal publication to Transactions of the International Society for Music Information Retrieval (TISMIR) is attached on the following page.
Dear Kristen Masada,

Thank you for submitting the manuscript, "Chord Recognition in Symbolic Music: A Segmental CRF Model, Segment-Level Features, and Comparative Evaluations on Classical and Popular Music" to Transactions of the International Society for Music Information Retrieval. With the online journal management system that we are using, you will be able to track its progress through the editorial process by logging in to the journal web site.

Your submission will be considered by our Editors. Submissions generally take 4-8 weeks undergo the review process. Following the completion of the review, you will be contacted by journal staff with review feedback.

You should be able to track your submission via your online account: https://transactions.ismir.net/jms/user/

Thank you for considering this journal as a venue for your work.

Kind regards,

________________________________________________________________________

Editorial Board Transactions of the International Society for Music Information Retrieval

http://transactions.ismir.net
8. Journal Submission

Attached on the next page is my journal submission to TISMIR.

ARTICLE TYPE

Chord Recognition in Symbolic Music: A Segmental CRF Model, Segment-Level Features, and Comparative Evaluations on Classical and Popular Music

Kristen Masada and Razvan Bunescu*

Abstract

We present a new approach to harmonic analysis that is trained to segment music into a sequence of chord spans tagged with a globally optimal set of chord labels. Formulated as a semi-Markov Conditional Random Field (semi-CRF), this joint segmentation and labeling approach enables the use of a rich set of segment-level features, such as segment purity and chord coverage, that capture the extent to which the events in an entire segment of music are compatible with a candidate chord label. The new chord recognition model is evaluated extensively on three corpora of classical music and a newly created corpus of rock music. Experimental results show that the semi-CRF model performs substantially better than previous approaches when trained on a sufficient number of labeled examples and remains competitive when the amount of training data is limited.

Keywords: harmonic analysis, chord recognition, semi-CRF, segmental CRF, symbolic music.

1. Introduction and Motivation

Harmonic analysis is an important step towards creating high level representations of tonal music. High level structural relationships form an essential component of music analysis, whose aim is to achieve a deep understanding of how music works. At its most basic level, harmonic analysis of music in symbolic form requires the partitioning of a musical input into segments along the time dimension, such that the notes in each segment correspond to a musical chord. This chord recognition task can often be time consuming and cognitively demanding, hence the utility of computer-based implementations. Reflecting historical trends in artificial intelligence, automatic approaches to harmonic analysis have evolved from purely grammar-based and rule-based systems (Winograd, 1968; Maxwell, 1992), to systems employing weighted rules and optimization algorithms (Temperley and Sleator, 1999; Pardo and Birmingham, 2002; Scholz and Ramalho, 2008; Rocher et al., 2009), to data driven approaches based on supervised machine learning (ML) (Raphael and Stoddard, 2003; Radicioni and Esposito, 2010). Due to their requirements for annotated data, ML approaches have also led to the development of music analysis datasets containing a large number of manually annotated harmonic structures, such as the 60 Bach Chorales introduced in Radicioni and Esposito (2010), and the 27 theme and variations of TAVERN (Devaney et al., 2015).

A relatively common strategy in ML approaches to chord recognition is to break the musical input into a sequence of short duration spans and then train sequence tagging algorithms such as Hidden Markov Models (HMMs) to assign a chord label to each span in the sequence (at the bottom in Figure 1). The spans can result from quantization using a fixed musical period such as half a measure (Raphael and Stoddard, 2003) or constructed from consecutive note onsets and offsets (Radicioni and Esposito, 2010). Variable-length chord segments are then created by joining consecutive spans labeled with the same chord symbol (at

*School of EECS, Ohio University, Athens, OH

Figure 1: Segment-based recognition (top) vs. event-based recognition (bottom), on measures 11 and 12 from Beethoven WoO68.
the top in Figure 1). A significant drawback of these short-span tagging approaches is that segments are not known during training and inference, therefore the ML model cannot use features that capture properties of segments that are known to be relevant with respect to their harmonic content. The chordal analysis system of Pardo and Birmingham (2002) is an example where the assignment of chords to segments takes into account segment-based features, however the features have pre-defined weights and it uses a processing pipeline where segmentation is done independently of chord labeling.

In this paper, we propose a machine learning approach to chord recognition formulated under the framework of semi-Markov Conditional Random Fields (semi-CRFs). Also called segmental CRFs, this class of probabilistic graphical models can be trained to do joint segmentation and labeling of symbolic music (Section 3), using efficient Viterbi-based inference algorithms whose time complexity is linear in the length of the input. Compared to sequence tagging approaches like HMMs or sequential CRFs, segmental CRFs enable the use of features that consider all the notes inside a segment. Correspondingly, we define a rich set of features that capture the extent to which the events in an entire segment of music are compatible with a candidate chord label (Section 4). The semi-CRF model incorporating these features is evaluated on three classical music datasets and a newly created dataset of popular music (Section 5). Experimental comparisons with two previous chord recognition models show that segmental CRFs obtain substantial improvements in performance on the three larger datasets, while also being competitive with the previous approaches on the smaller dataset (Section 6).

2. Types of Chords in Tonal Music
A chord is a group of notes that form a cohesive harmonic unit to the listener when sounding simultaneously (Aldwell et al., 2011). We design our system to handle three major categories of chords: triads, augmented 6th chords, and suspended and power chords.

2.1 Triads
A triad is the prototypical instance of a chord. It is based on a root note, which forms the lowest note of the chord. A third and a fifth are then built on top of this root to create a three-note chord. The quality of the third and fifth intervals determines the mode of a triad. For our system, we consider three triad modes: major, minor, and diminished. A major triad consists of a major third interval (i.e. 4 half steps) between the root and third, as well as a perfect fifth (7 half steps) between the root and fifth. A minor triad swaps the major third for a minor third interval (3 half steps) between the third and fifth. Lastly, a diminished triad maintains the minor third, but possesses a diminished fifth (6 half steps) between the root and fifth.

A triad can contain an added note, or a fourth note. We include three possible added notes in our system: a fourth, a sixth, and a seventh. A fourth chord contains an interval of a perfect fourth (5 half steps) between the root and the added note for all modes. In contrast, the interval between the root and added note of a sixth chord of any mode is a major sixth (9 half steps). For seventh chords, the added note interval varies. If the triad is major, the added note frequently forms a major seventh (11 half steps) with the root, called a major seventh chord. It can also form a minor seventh (10 half steps) to create a dominant seventh chord. For our system, we classify both as major seventh chords, given that the underlying triad is major. If the triad is minor, the added seventh can again either form an interval of a major seventh, creating a minor-major seventh chord, or a minor seventh, forming a minor seventh chord. We consider both as minor seventh chords. Finally, diminished triads most frequently contain a diminished seventh interval (9 half steps), producing a fully diminished seventh chord, or a minor seventh interval, creating a half-diminished seventh chord. For simplicity, we label both as diminished seventh chords.

2.2 Augmented 6th Chords
Augmented 6th chords are a type of chromatic chord defined by an augmented sixth interval between the lowest and highest notes of the chord (Aldwell et al., 2011). Typically the augmented sixth interval resolves to an octave on the fifth scale degree to uniquely switch to dominant harmony. The three most common types of augmented 6th chords are Italian, German, and French sixth chords.

A significant drawback of these short-span tagging approaches is that segments are not known during training and inference, therefore the ML model cannot use features that capture properties of segments that are known to be relevant with respect to their harmonic content. The chordal analysis system of Pardo and Birmingham (2002) is an example where the assignment of chords to segments takes into account segment-based features, however the features have pre-defined weights and it uses a processing pipeline where segmentation is done independently of chord labeling.

In this paper, we propose a machine learning approach to chord recognition formulated under the framework of semi-Markov Conditional Random Fields (semi-CRFs). Also called segmental CRFs, this class of probabilistic graphical models can be trained to do joint segmentation and labeling of symbolic music (Section 3), using efficient Viterbi-based inference algorithms whose time complexity is linear in the length of the input. Compared to sequence tagging approaches like HMMs or sequential CRFs, segmental CRFs enable the use of features that consider all the notes inside a segment. Correspondingly, we define a rich set of features that capture the extent to which the events in an entire segment of music are compatible with a candidate chord label (Section 4). The semi-CRF model incorporating these features is evaluated on three classical music datasets and a newly created dataset of popular music (Section 5). Experimental comparisons with two previous chord recognition models show that segmental CRFs obtain substantial improvements in performance on the three larger datasets, while also being competitive with the previous approaches on the smaller dataset (Section 6).

2. Types of Chords in Tonal Music
A chord is a group of notes that form a cohesive harmonic unit to the listener when sounding simultaneously (Aldwell et al., 2011). We design our system to handle three major categories of chords: triads, augmented 6th chords, and suspended and power chords.

2.1 Triads
A triad is the prototypical instance of a chord. It is based on a root note, which forms the lowest note of the chord. A third and a fifth are then built on top of this root to create a three-note chord. The quality of the third and fifth intervals determines the mode of a triad. For our system, we consider three triad modes: major, minor, and diminished. A major triad consists of a major third interval (i.e. 4 half steps) between the root and third, as well as a perfect fifth (7 half steps) between the root and fifth. A minor triad swaps the major third for a minor third interval (3 half steps) between the third and fifth. Lastly, a diminished triad maintains the minor third, but possesses a diminished fifth (6 half steps) between the root and fifth.

A triad can contain an added note, or a fourth note. We include three possible added notes in our system: a fourth, a sixth, and a seventh. A fourth chord contains an interval of a perfect fourth (5 half steps) between the root and the added note for all modes. In contrast, the interval between the root and added note of a sixth chord of any mode is a major sixth (9 half steps). For seventh chords, the added note interval varies. If the triad is major, the added note frequently forms a major seventh (11 half steps) with the root, called a major seventh chord. It can also form a minor seventh (10 half steps) to create a dominant seventh chord. For our system, we classify both as major seventh chords, given that the underlying triad is major. If the triad is minor, the added seventh can again either form an interval of a major seventh, creating a minor-major seventh chord, or a minor seventh, forming a minor seventh chord. We consider both as minor seventh chords. Finally, diminished triads most frequently contain a diminished seventh interval (9 half steps), producing a fully diminished seventh chord, or a minor seventh interval, creating a half-diminished seventh chord. For simplicity, we label both as diminished seventh chords.

2.2 Augmented 6th Chords
Augmented 6th chords are a type of chromatic chord defined by an augmented sixth interval between the lowest and highest notes of the chord (Aldwell et al., 2011). Typically the augmented sixth interval resolves to an octave on the fifth scale degree to uniquely switch to dominant harmony. The three most common types of augmented 6th chords are Italian, German, and French sixth chords.
a sharpened root and an added minor seventh interval. We refer to this additional note as a fifth, given that it forms a perfect fifth with the flattened sixth.

French sixth chords also possess the same notes as Italian sixth chords, but with an added second scale degree. Thus, they are diminished II chords with a sharpened third and added minor seventh in second inversion. We again refer to the added note as a fifth, though it forms a diminished fifth with the flattened sixth this time.

2.3 Suspended and Power Chords

Both suspended and power chords are similar to triads in that they contain a root and a perfect fifth. They differ, however, in their omission of the third. Suspended second chords use a second as replacement for this third, forming a major second (2 half steps) with the root, while suspended fourth chords employ a perfect fourth as replacement (Taylor, 1989). Both the suspended second and fourth often resolve to a more stable third. In addition to these two kinds of suspended chords, our system considers suspended fourth chords that contain an added minor seventh, forming a dominant seventh suspended fourth chord.

![Figure 4: Suspended and power chords.](image)

In contrast with suspended chords, power chords do not contain a replacement for the missing third. They simply consist of a root and a perfect fifth. Though they are not formally considered to be chords in classical music, they are commonly referred to in both rock and pop music (Denyer, 1992).

3. Semi-CRF Model for Chord Recognition

Since harmonic changes may occur only when notes begin or end, we first create a sorted list of all the note onsets and offsets in the input music, i.e., the list of partition points (Pardo and Birmingham, 2002). A basic music event (Radicioni and Esposito, 2010) is then defined as the set of notes sounding in the time interval between two consecutive partition points. Let \( s = (s_1, s_2, \ldots, s_K) \) denote a segmentation of the musical input \( x \), where a segment \( s_k = (s_{k-1}, s_k, l_k) \) is identified by the indices \( s_{k-1}, s_k, l_k \) of its first and last events, respectively.

Let \( y = (y_1, y_2, \ldots, y_K) \) be the vector of chord labels corresponding to the segmentation \( s \). The set of labels can range from coarse grained labels that indicate only the chord root (Temperley and Sleator, 1999) to fine grained labels that capture mode, inversions, added and missing notes (Harte, 2010), and even chord function (Devaney et al., 2015). Here we follow the middle ground proposed by Radicioni and Esposito (Radicioni and Esposito, 2010) and define a set of labels that encode the chord root (12 pitch classes), the mode (major, minor, diminished), and the added note (none, fourth, sixth, seventh), for a total of 144 different labels. Since the labels do not encode for function, the model does not require knowing the key in which the input was written.

A semi-Markov CRF (Sarawagi and Cohen, 2004) defines a probability distribution over segmentations and their labels as shown in Equations 1 and 2. Here, the global segmentation feature vector \( F \) decomposes as a sum of local segment feature vectors \( f(s_k, y_k, y_{k-1}, x) \), with label \( y_0 \) set to a constant “no chord” value. The ensuing factorization of the distribution enables an efficient computation of the most likely segmentation \( \text{argmax}_y P(s, y|x, w) \) using a semi-Markov analogue of the Viterbi algorithm (Sarawagi and Cohen, 2004).

\[
P(s, y|x, w) = \frac{e^{w^T F(s, y, x)}}{Z(x)} \tag{1}
\]

\[
Z(x) = \sum_{s, y} e^{w^T F(s, y, x)}
\]

\[
F(s, y, x) = \sum_{k=1}^{K} f(s_k, y_k, y_{k-1}, x) \tag{2}
\]

Following Muis and Lu (Muis and Lu, 2016), for faster inference, we further restrict the local segment features to two types: segment-label features \( f(s_k, y_k, x) \) that depend on the segment and its label, and label transition features \( g(y_k, y_{k-1}, x) \) that depend on the labels of the current and previous segments. The corresponding probability distribution over segmentations is shown in Equations 3 to 5 below.

Given an arbitrary segment \( s \) and a label \( y \), the vector of segment-label features can be written as \( f(s, y, x) = [f_1(s, y), \ldots, f_{12}(s, y)] \), where the input \( x \) is left implicit in order to compress the notation. Similarly, given arbitrary labels \( y \) and \( y’ \), the vector of label transition features can be written as \( g(y, y’, x) = [g_1(y, y’), \ldots, g_{12}(y, y’)] \). In Section 4 we describe the set of segment-label features \( f_i(s, y) \) and label transition features \( g_i(y, y’) \) that are used in our semi-CRF chord recognition system.

\[
P(s, y|x, w) = \frac{e^{w^T F(s, y, x) + u^T G(s, y, x)}}{Z(x)} \tag{3}
\]

\[
F(s, y, x) = \sum_{k=1}^{K} f(s_k, y_k, x) \tag{4}
\]

\[
G(s, y, x) = \sum_{k=1}^{K} g(y_k, y_{k-1}, x) \tag{5}
\]

As probabilistic graphical models, semi-CRFs can be represented using factor graphs. Factor graphs (Kschischang et al., 2001) are bipartite graphs that express how a global function of many variables, such as \( P(s, y|x, w) \) from Equation 3, factorizes into a product of
local functions defined over fewer variables, such as $f$ and $g$ in Equations 4 and 5. Figure 5 shows the factor graph template corresponding to the semi-CRF.

Figure 5: Factor graph representation of the semi-CRF.

4. Chord Recognition Features

Given a segment $s$ and chord $y$, we will use the following notation:

- $s.Notes$, $s.N$ = the set of notes in the segment $s$.
- $s.Events$, $s.E$ = the sequence of events in $s$.
- $e.len$, $n.len$ = the length of event $e$ or note $n$, in quarters.
- $e.acc$, $n.acc$ = the accent value of event $e$ or note $n$, as computed by the beatStrength() function in Music21. This is a value that is determined based on the metrical position of $n$, e.g. in a song written in a 4/4 time signature, the first beat position would have a value of 1.0, the third beat 0.5, and the second and fourth beats 0.25. Any other eighth note position within a beat would have a value of 0.125. Any sixteenth note position strictly within the beat would have a value of 0.0625, and so on.
- $y.root$, $y.third$, and $y.fifth$ = the triad tones of the chord $y$.
- $y.added$ = the added note of chord $y$, if $y$ is an added tone chord.

We designed a set of heuristics to determine whether a note $n$ from a segment $s$ is a figuration note with respect to a candidate chord label $y$. The heuristic rules shown below discover four types of figurations: passing and neighbor notes (Figure 6), suspensions and anticipations (Figure 7).

**Passing:** There are two anchor notes $n_1$ and $n_2$ such that: $n_1$’s offset coincides with $n$’s onset; $n_2$’s onset coincides with $n$’s offset; $n_1$ is one scale step below $n$ and $n_2$ is one step above $n$, or $n_1$ is one step above $n$ and $n_2$ one step below; $n$ is not longer than either $n_1$ or $n_2$; the accent value of $n$ is strictly smaller than the accent value of $n_1$; at least one of the two anchor notes belongs to segment $s$; $n$ is non-harmonic with respect to chord $y$, i.e. $n$ is not equivalent to the root, third, fifth, or added note of $y$; both $n_1$ and $n_2$ are harmonic with respect to the segments they belong to.

**Suspension:** Note $n$ belongs to the first event of segment $s$. There is an anchor note $m$ in the previous event (last event in the previous segment) such that: $m$ and $n$ have the same pitch; $n$ is either tied with $m$ (i.e. held over) or $m$’s offset coincides with $n$’s onset (i.e. restruck); $n$ is not longer than $m$; $n$ is non-harmonic with respect to chord $y$, while $m$ is harmonic with respect to the previous chord.

**Anticipation:** Note $n$ belongs to the last event of segment $s$. There is an anchor note $m$ in the next event (first event in the next segment) such that: $n$ and $m$ have the same pitch; $m$ is either tied with $n$ (i.e. held over) or $n$’s offset coincides with $m$’s onset (i.e. restruck); $n$ is not longer than $m$; $n$ is non-harmonic with respect to chord $y$, while $m$ is harmonic relative to all other notes in its event.

Furthermore, because we are using the weak semi-CRF features shown in Equation 4, we need a heuristic to determine whether an anchor note is harmonic whenever the anchor note precedes the current segment. The heuristic simply looks at the other notes in the event containing the anchor note: if the event contains 2 or more other notes, at least 2 of them need to be consonant with the anchor, i.e. intervals of octaves.
fifths, thirds, and their inversions; if the event contains just one other note, it has to be consonant with the anchor.

We emphasize that the rules mentioned above for detecting figuration notes are only approximations. We recognize that correctly identifying figuration notes can also depend on subtler stylistic and contextual cues, thus allowing for exceptions to each of these rules.

Equipped with this heuristic definition of figuration notes, we augment the notation as follows:

- \( s.\text{Fig}(y) \) = the set of notes in \( s \) that are figuration with respect to chord \( y \).
- \( s.\text{NonFig}(y) = s.\text{Notes} - s.\text{Fig}(y) \) = the set of notes in \( s \) that are not figuration with respect to \( y \).

Some of the segment-label features introduced in this section have real values. Given a real-valued feature \( f(s, y) \) that takes values in \([0, 1]\), we discretize it into \( K + 2 \) Boolean features by partitioning the \([0, 1]\) interval into a set of \( K \) subinterval bins \( \mathcal{B} = \{ (b_{k-1}, b_k) | 1 \leq k \leq K \} \). For each bin, the corresponding Boolean feature determines whether \( f(s, y) \in (b_{k-1}, b_k) \). Additionally, two Boolean features are defined for the boundary cases \( f(s, y) = 0 \) and \( f(s, y) = 1 \). For each real-valued feature, unless specified otherwise, we use the bin set \( \mathcal{B} = \{ 0, 0.1, ..., 0.9, 1.0 \} \).

We propose five major types of features. Segment purity features compute the percentage of segment notes that belong to a given chord (Section 4.1). We include these on the grounds that segments with a higher purity with respect to a chord are more likely to be labeled with that chord. Chord coverage features determine if each note in a given chord appears at least once in the segment (Section 4.2). Similar to segment purity, if the segment cover a higher percentage of the chord’s notes, it is more likely to be labeled with that chord. Bass features determine which note of a given chord appears as bass in the segment (Section 4.3). For a correctly labeled segment, its bass note often matches the root of its chord label. If the bass note instead matches the chord’s third, fifth, or an added dissonance, this may indicate that the chord \( y \) is inverted or incorrect. Chord bigram features capture chord transition information (Section 4.4). These features are useful in that the arrangement of chords in chord progressions is an important component of harmonic syntax. Finally, we include metrical accent features for chord changes, as chord segments are more likely to begin on accented beats (Section 4.5).

4.1 Segment Purity

The segment purity feature \( f_1(s, y) \) computes the fraction of the notes in segment \( s \) that are harmonic, i.e. belong to chord \( y \):

\[
f_1(s, y) = \frac{\sum_{n \in s.\text{Notes}} 1(n \in y)}{|s.\text{Notes}|}
\]

The duration-weighted version \( f_2(s, y) \) of the purity feature weighs each note \( n \) by its length \( n.\text{len} \):

\[
f_2(s, y) = \frac{\sum_{n \in s.\text{Notes}} 1(n \in y) \times n.\text{len}}{\sum_{n \in s.\text{Notes}} n.\text{len}}
\]

The accent-weighted version \( f_3(s, y) \) of the purity feature weighs each note \( n \) by its accent weight \( n.\text{acc} \):

\[
f_3(s, y) = \frac{\sum_{n \in s.\text{Notes}} 1(n \in y) \times n.\text{acc}}{\sum_{n \in s.\text{Notes}} n.\text{acc}}
\]

The 3 real-valued features are discretized using the default bin set \( \mathcal{B} \).

4.1.1 Figuration-Controlled Segment Purity

For each segment purity feature, we create a figuration-controlled version that ignores notes that were heuristically detected as figuration, i.e. replace \( s.\text{Notes} \) with \( s.\text{NonFig}(y) \) in each feature formula.

4.2 Chord Coverage

The chord coverage features determine which of the chord notes belong to the segment. In this section, each of the coverage features are non-zero only for major, minor, and diminished triads and their added note counterparts. This is implemented by first defining an indicator function \( y.\text{Triad} \) that is 1 only for triads and added note chords, and then multiplying it into all the triad features from this section.

\[
y.\text{Triad} = 1[y.\text{mode} \in \{ \text{maj, min, dim} \}]
\]

Furthermore, we compress notation by showing the mode predicates as attributes of the label, e.g. \( y.\text{maj} \) is a predicate equivalent with testing whether \( y.\text{mode} = \text{maj} \). Thus, an equivalent formulation of \( y.\text{Triad} \) is as follows:

\[
y.\text{Triad} = 1[y.\text{maj} \lor y.\text{min} \lor y.\text{dim}]
\]

To avoid clutter, we do not show \( y.\text{Triad} \) in any of the features below, although it is assumed to be multiplied into all of them. The first 3 coverage features refer to the triad notes:

\[
f_4(s, y) = 1[y.\text{root} \in s.\text{Notes}]
\]
\[
f_5(s, y) = 1[y.\text{third} \in s.\text{Notes}]
\]
\[
f_6(s, y) = 1[y.\text{fifth} \in s.\text{Notes}]
\]

A separate feature determines if the segment contains all the notes in the chord:

\[
f_7(s, y) = \prod_{n \in y} 1(n \in s.\text{Notes})
\]

A chord may have an added tone \( y.\text{added} \), such as a 4th, a 6th, or a 7th. If a chord has an added tone, we
define two features that determine whether the segment contains the added note:

\[ f_8(s, y) = \sum_{n \in s.\text{Notes}} 1\left[\exists y.\text{added} \land y.\text{added} \in s.\text{Notes}\right] \]
\[ f_9(s, y) = \sum_{n \in s.\text{Notes}} 1\left[\exists y.\text{added} \land y.\text{added} \in s.\text{Notes}\right] \]

Through the first feature, the system can learn to prefer the added tone version of the chord when the segment contains it, while the second feature enables the system to learn to prefer the triad-only version if no added tone is in the segment. To prevent the system from recognizing added chords too liberally, we add a feature that is triggered whenever the total length of the added note in the segment is greater than the total length of the root:

\[ a.len(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{added}\right] \cdot n.\text{len} \]
\[ r.len(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{root}\right] \cdot n.\text{len} \]
\[ f_{10}(s, y) = \sum_{n \in s.\text{Notes}} 1\left[\exists y.\text{added} \land y.\text{added} \in s.\text{Notes}\right] \cdot \left[a.len(s, y) > r.len(s, y)\right] \]

The duration-weighted versions of the chord coverage features weigh each chord tone by its total duration in the segment. For the root, the feature would be computed as shown below:

\[ f_{11}(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{root}\right] \cdot n.\text{len} \]

Similar features \( f_{12} \) and \( f_{13} \) are computed for the third and the fifth. The corresponding accent-weighted features \( f_{14}, f_{15} \), and \( f_{16} \) are computed in a similar way, by replacing the note duration \( n.\text{len} \) in the duration-weighted formulas with the note accent value \( n.\text{acc} \).

The duration-weighted feature for the added tone is computed similarly:

\[ f_{17}(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{added}\right] \cdot n.\text{len} \]

Furthermore, by replacing \( n.\text{len} \) with \( n.\text{acc} \), we also obtain the accent-weighted version \( f_{18} \).

An alternative definition of duration-weighted features is based on the proportion of the segment time that is covered by a particular chord note. The corresponding duration-weighted feature for the chord root is shown below:

\[ f_{19}(s, y) = \sum_{e \in s.\text{Events}} 1\left[y.\text{root} \in e\right] \cdot e.\text{len} \]

Similar duration-weighted features normalized by the segment length are defined for thirds, fifths, and added notes.

All duration-weighted and accent-weighted features are discretized using the default bin set \( \mathcal{B} \).

### 4.2.1 Chord Coverage for Augmented 6th Chords

The features defined in this section are non-zero only for augmented 6th chord labels. Similar to before, we define an indicator function \( y.\text{AS} \) that is 1 only for augmented 6th chords and implicitly multiply this into each of the features from this section.

\[ y.\text{AS} = 1\left[y.\text{mode} \in \{\text{fr6}, \text{fr6}, \text{ger6}\}\right] \]
\[ y.\text{AS} = 1\left[y.\text{fr6} \lor y.\text{ger6}\right] \]

We define an additional indicator function \( y.\text{FG} \) that is 1 only for French and German 6th chords.

\[ y.\text{FG} = 1\left[y.\text{fr6} \lor y.\text{ger6}\right] \]

The coverage features for augmented 6th chords are overall analogous to the ones for triad chords.

\[ a.s_1(s, y) = 1\left[y.\text{bass} \in s.\text{Notes}\right] \]
\[ a.s_2(s, y) = 1\left[y.\text{3rd} \in s.\text{Notes}\right] \]
\[ a.s_3(s, y) = 1\left[y.\text{6th} \in s.\text{Notes}\right] \]
\[ a.s_4(s, y) = 1\left[y.\text{FG} \land y.\text{5th} \in s.\text{Notes}\right] \]

The duration-weighted versions are as follows:

\[ a.s_5(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{bass}\right] \cdot n.\text{len} \]
\[ a.s_6(s, y) = \sum_{n \in s.\text{Notes}} 1\left[n = y.\text{5th}\right] \cdot n.\text{len} \]

As before, we replace \( n.\text{len} \) with \( n.\text{acc} \) to obtain the accent-weighted version of \( a.s_5 \) and \( a.s_6 \). We also define segment-based duration-weighted features:

\[ a.s_7(s, y) = \sum_{e \in s.\text{Events}} 1\left[y.\text{bass} \in e\right] \cdot e.\text{len} \]

### 4.2.2 Chord Coverage for Suspended and Power Chords

As before, we define the features in this section to be non-zero only for suspended or power chord labels by implicitly multiplying them with an indicator function \( y.\text{SP} \).

\[ y.\text{SP} = 1\left[y.\text{sus2} \lor y.\text{sus4} \lor y.7\text{sus4} \lor y.\text{pow}\right] \]

The coverage features for suspended and power chords are also similar to the ones defined for triad chords.

\[ s.p_1(s, y) = 1\left[y.\text{root} \in s.\text{Notes}\right] \]
\[ s.p_2(s, y) = 1\left[y.\text{sus2} \land y.2\text{nd} \in s.\text{Notes} \lor (y.\text{sus4} \lor y.7\text{sus4}) \land y.\text{4th} \in s.\text{Notes}\right] \]
\[ s.p_3(s, y) = 1\left[y.\text{5th} \in s.\text{Notes}\right] \]
\[ s.p_4(s, y) = 1\left[y.7\text{sus4} \land y.7\text{th} \in s.\text{Notes}\right] \]
\[ s.p_5(s, y) = 1\left[y.7\text{sus4} \land y.7\text{th} \in s.\text{Notes}\right] \]
The duration-weighted versions are as follows:

\[
sp_7(s, y) = \frac{rlen(s, y)}{\text{n.len}} \\
sp_8(s, y) = 1[y.7sus4 \land \frac{alen(s, y)}{\text{n.len}}]
\]

We also define accent-weighted versions of \(sp_7\) and \(sp_8\), as well as segment-based duration-weighted features:

\[
sp_9(s, y) = \sum_{\text{e} \in \text{e.Events}} 1[\text{e.root} \in e] \cdot \text{e.len}
\]

### 4.2.3 Figuration-Controlled Chord Coverage

For each chord coverage feature, we create a figuration-controlled version that ignores notes that were heuristically detected as figuration, i.e. replace \(s.\text{Notes}\) with \(s.\text{NonFig}(y)\) in each feature formula.

### 4.3 Bass

The bass note provides the foundation for the harmony of a musical segment. For a correct segment, its bass note often matches the root of its chord label. If the bass note instead matches the chord's third, fifth, or added dissonance, this may indicate that the chord is inverted. Thus, comparing the bass note with the chord tones can provide useful features for determining whether a segment is compatible with a chord label. As in Section 4.2, we implicitly multiply each of these features with \(y.\text{Triad}\) so that they are non-zero only for triads and added note chords.

There are multiple ways to define the bass note of a segment \(s\). One possible definition is the lowest note of the first event in the segment, i.e. \(s.e_1.\text{bass}\). Comparing it with the root, third, fifth, and added tones of a chord results in the following features:

\[
\begin{align*}
f_{20}(s, y) &= 1[s.e_1.\text{bass} = y.\text{root}] \\
f_{21}(s, y) &= 1[s.e_1.\text{bass} = y.\text{third}] \\
f_{22}(s, y) &= 1[s.e_1.\text{bass} = y.\text{fifth}] \\
f_{23}(s, y) &= 1[\exists \text{y.added} \land s.e_1.\text{bass} = y.\text{added}]
\end{align*}
\]

An alternative definition of the bass note of a segment is the lowest note in the entire segment, i.e. \(\text{min}_{e \in E} e.\text{bass}\). The corresponding features will be:

\[
\begin{align*}
f_{24}(s, y) &= 1[y.\text{root} = \text{min} e.\text{bass}] \\
f_{25}(s, y) &= 1[y.\text{third} = \text{min} e.\text{bass}] \\
f_{26}(s, y) &= 1[y.\text{fifth} = \text{min} e.\text{bass}] \\
f_{27}(s, y) &= 1[\exists \text{y.added} \land y.\text{added} = \text{min} \ e.\text{bass}] \\
\end{align*}
\]

The duration-weighted versions of the bass features weight each chord tone by the time it is used as the lowest note in each segment event, normalized by the duration of the bass notes in all the events. For the root, the feature is computed as shown below:

\[
f_{28}(s, y) = \frac{\sum_{e \in \text{e.Events}} 1[e.\text{bass} = y.\text{root}] \cdot \text{e.len}}{\sum_{e \in \text{e.Events}} \text{e.len}}
\]

Similar features \(f_{29}\) and \(f_{30}\) are computed for the third and the fifth. The duration-weighted feature for the added tone is computed as follows:

\[
f_{31}(s, y) = \frac{\sum_{e \in \text{e.Events}} 1[e.\text{bass} = y.\text{root}] \cdot \text{e.len}}{\sum_{e \in \text{e.Events}} \text{e.len}}
\]

The corresponding accent-weighted features \(f_{32}, f_{33}, f_{34}\) are computed in a similar way, by replacing the bass duration \(e.\text{bass}.\text{len}\) in the duration-weighted formulas with the note accent value \(e.\text{bass}.\text{acc}\).

All duration-weighted and accent-weighted features are discretized using the default bin set \(\mathcal{B}\).

### 4.3.1 Bass Features for Augmented 6th Chords

Similar to the chord coverage features in Section 4.2.1, we assume that the indicator \(y.\text{AS}\) is multiplied into all features in this section, which means they are non-zero only for augmented 6th chords.

\[
\begin{align*}
\text{as}_{9}(s, y) &= 1[s.e_1.\text{bass} = y.\text{bass}] \\
\text{as}_{9}(s, y) &= 1[s.e_1.\text{bass} = y.3rd] \\
\text{as}_{10}(s, y) &= 1[s.e_1.\text{bass} = y.6th] \\
\text{as}_{11}(s, y) &= 1[y.f r6 \lor y.ger6] \land s.e_1.\text{bass} = y.5th] \\
\text{as}_{12}(s, y) &= 1[y.\text{bass} = \text{min} e.\text{bass}] \\
\text{as}_{13}(s, y) &= 1[y.3rd = \text{min} e.\text{bass}] \\
\text{as}_{14}(s, y) &= 1[y.6th = \text{min} e.\text{bass}] \\
\text{as}_{15}(s, y) &= 1[y.FG \land y.5th = \text{min} e.\text{bass}] \\
\end{align*}
\]

We define the following duration-weighted version for the augmented sixth bass and fifth.

\[
\begin{align*}
\text{as}_{16}(s, y) &= \frac{\sum_{e \in \text{e.Events}} 1[e.\text{bass} = y.\text{bass}] \cdot \text{e.len}}{\sum_{e \in \text{e.Events}} \text{e.len}} \\
\text{as}_{17}(s, y) &= 1[y.FG] \cdot \frac{\sum_{e \in \text{e.Events}} 1[e.\text{bass} = y.5th] \cdot \text{e.len}}{\sum_{e \in \text{e.Events}} \text{e.len}}
\end{align*}
\]
4.3.2 Bass Features for Suspended and Power Chords
The indicator \( y.SP \) is multiplied into all features in this section like in Section 4.2.2, meaning they are non-zero only for suspended and power chords.

\[
s_p(3, y) = 1 \{ \text{s.e}_1 \text{bass} = y \text{root} \}
\]
\[
s_p(4, y) = 1 \{ y \text{sus}2 \wedge \text{s.e}_1 \text{bass} = y \text{2nd} \} \vee (y \text{sus}4 \wedge y \text{7sus}4) \wedge \text{s.e}_1 \text{bass} = y \text{4th}
\]
\[
s_p(5, y) = 1 \{ \text{s.e}_1 \text{bass} = y \text{5th} \}
\]
\[
s_p(6, y) = 1 \{ y \text{7sus}4 \wedge \text{s.e}_1 \text{bass} = y \text{7th} \}
\]
\[
s_p(14, y) = 1 \{ y \text{root} = \min \text{e.bass} \}
\]
\[
s_p(15, y) = 1 \{ y \text{sus}2 \wedge y \text{2nd} = \min \text{e.bass} \} \vee (y \text{sus}4 \vee y \text{7sus}4) \wedge y \text{4th} = \min \text{e.bass}
\]
\[
s_p(16, y) = 1 \{ y \text{5th} = \min \text{e.bass} \}
\]
\[
s_p(17, y) = 1 \{ y \text{7sus}4 \wedge y \text{7th} = \min \text{e.bass} \}
\]

The duration-weighted version for the root and seventh are computed as follows:

\[
s_p(18, y) = \frac{\sum_{e \in \mathcal{E}} 1 \{ \text{e.bass} = y \text{root} \} \cdot e \text{len}}{\sum_{e \in \mathcal{E}} e \text{len}}
\]
\[
s_p(19, y) = 1 \{ y \text{7sus}4 \} \cdot \frac{\sum_{e \in \mathcal{E}} e \text{len}}{\sum_{e \in \mathcal{E}} e \text{len}}
\]

4.3.3 Figuration-Controlled Bass
For each bass feature, we create a figuration-controlled version that ignores event bass notes that were heuristically detected as figuration, i.e. replace \( e \in \text{s.Event} \) with \( e \in \text{s.Event} \wedge \text{e.bass} \notin \text{s.Fig}(y) \) in each feature formula.

4.4 Chord Bigrams
The arrangement of chords in chord progressions is an important component of harmonic syntax (Aldwell et al., 2011). A first order semi-Markov CRF model can capture chord sequencing information only through the chord labels \( y \) and \( y' \) of the current and previous segment. To obtain features that generalize to unseen chord sequences, we follow Radicioni and Esposito (2010) and create chord bigram features using only the \( \text{mode} \), the \( \text{added} \) note, and the interval in semitones between the roots of the two chords. We define the possible modes of a chord label as follows:

\[\mathcal{M} = \{ \text{maj}, \text{min}, \text{dim} \}\]
\[\cup \{ \text{it6}, \text{fr6}, \text{ger6} \}\]
\[\cup \{ \text{sus}2, \text{sus}4, 7\text{sus}4, \text{pow} \}\]

Other than the common major (maj), minor (min), and diminished (dim) modes, the following chord types have been included in \( \mathcal{M} \) as modes:

- Augmented 6th chords: Italian 6th (it6), French 6th (fr6), and German 6th (ger6).
- Suspended chords: suspended second (sus2), suspended fourth (sus4), dominant seventh suspended fourth (7sus4).
- Power (pow) chords.

Correspondingly, the chord bigrams can be generated using the feature template below:

\[
g_1(y, y') = 1 \{ (y \text{mode}, y' \text{mode}) \in \mathcal{M} \times \mathcal{M} \}
\]
\[
\wedge (y \text{added}, y' \text{added}) \in \{ \varnothing, 4, 6, 7 \} \times \{ \varnothing, 4, 6, 7 \}
\]
\[
\wedge |y \text{root} - y' \text{root}| = 0, 1, ..., 11] \]

Note that \( y \text{root} \) is replaced with \( y \text{bass} \) for augmented 6th chords. Additionally, \( y \text{added} \) is always none (\( \varnothing \)) for augmented 6th, suspended, and power chords. Thus, \( g_1(y, y') \) is a feature template that can generate (3 triad modes \( \times 4 \) added + 3 aug6 modes + 3 sus modes + 1 pow mode)\(^2 \times 12 \) intervals = 4,332 distinct features. To reduce the number of features, we use only the \( (\text{mode.added} - \text{mode.added})^1 \) - interval combinations that appear in the manually annotated chord bigrams from the training data.

4.5 Chord Changes and Metrical Accent
In general, repeating a chord creates very little accent, whereas changing a chord tends to attract an accent (Aldwell et al., 2011). Although conflict between meter and harmony is an important compositional resource, in general chord changes support the meter. Correspondingly, a new feature is defined as the accent value of the first event in a candidate segment:

\[
f_3(5, y) = \text{s.e}_1 \text{acc}
\]

5. Chord Recognition Datasets
For evaluation, we used four chord recognition datasets:

1. BaCh: this is the Bach Choral Harmony Dataset, a corpus of 60 four-part Bach chorales that contains 5,664 events and 3,090 segments in total (Radicioni and Esposito, 2010).
2. TAVERN: this is a corpus of 27 complete sets of theme and variations for piano, composed by Mozart and Beethoven. It consists of 63,876 events and 12,802 segments overall (Devaney et al., 2015).
3. KP Corpus: the Kostka-Payne corpus is a dataset of 46 excerpts compiled by Bryan Pardo from Kostka and Payne’s music theory textbook. It contains 3,888 events and 911 segments (Kostka and Payne, 1984).
4. Rock: this is a corpus of 59 pop and rock songs that we compiled from Hal Leonard’s The Best Rock Songs Ever (Easy Piano) songbook. It is 25,621 events and 4,221 segments in length.
contain chords with added fourth or sixth notes. How-

Like TAVERN, the Kostka-Payne (KP) corpus labels, of which 69 appear in the dataset. 

5.3 The Kostka and Payne Corpus

The manual annotation are enharmonic, e.g. C-sharp major and D-flat major, or D-sharp major and E-flat major. Reliably producing one of two enharmonic chords cannot be expected from a system that is agnostic of the key context. Therefore, we normalize the chord labels and for each mode we define a set of 12 canonical roots, one for each scale degree. When two enharmonic chords are available for a given scale degree, we selected the one with the fewest sharps or flats in the corresponding key signature. Consequently, for the major mode we use the canonical root set \{C, Db, D, Eb, G, Ab, A, Bb, B\}, whereas for the minor and diminished modes we used the root set \{C, C#, D, D#, F, F#, G, G#, A, Bb, B\}. Thus, if a chord is manually labeled as C-sharp major, the label is automatically changed to the enharmonic D-flat major.

The actual chord notes used in the music are left unchanged. Whether they are spelled with sharps or flats is immaterial, as long as they are enharmonic with the root, third, fifth, or added note of the labeled chord.

5.2 The TAVERN Dataset

The TAVERN dataset currently contains 17 works by Beethoven (181 variations) and 10 by Mozart (100 variations). The themes and variations are divided into a total of 1,060 phrases, 939 in major and 121 in minor. The pieces have two levels of segmentations: chords and phrases. The chords are annotated with Roman numerals, using the Humdrum representation for functional harmony. When finished, each phrase will have annotations from two different experts, with a third expert adjudicating cases of disagreement between the two. After adjudication, a unique annotation of each phrase is created and joined with the note data into a combined file encoded in standard **kern format. However, many pieces do not currently have the second annotation or the adjudicated version. Consequently, we only used the first annotation for each of the 27 sets. Furthermore, since our chord recognition approach is key agnostic, we developed a script that automatically translated the Roman numeral notation into the key-independent canonical set of labels used in BaCh. Because the TAVERN annotation does not mark added fourth or sixth notes, the only added chords that were generated by the translation script were those containing sevenths. This results in a set of 72 possible labels, of which 69 appear in the dataset.

5.3 The Kostka and Payne Corpus

Like TAVERN, the Kostka-Payne (KP) corpus does not contain chords with added fourth or sixth notes. However, it includes fine-grained chord types that are outside of our current label set, such as fully and half diminished seventh chords, dominant seventh chords, and dominant seventh flat ninth chords. We map these seventh chord variants to the corresponding diminished and major seventh chord labels, as discussed in Section 2. Chords with ninth intervals are mapped to the corresponding chord without the ninth in our label set. The KP Corpus also contains the three types of augmented 6th chords introduced in Section 2. Thus, by extending our chord set to include augmented 6th labels, there are 12 roots × 3 triad modes × 2 added notes + 12 bass notes × 3 aug6 modes = 108 possible labels overall. Of these, 76 appear in the dataset.

A number of MIDI files in the KP corpus contain unlabeled sections at the beginning of the song. These sections also appear as unlabeled in the original Kostka-Payne textbook. We omitted these sections from our evaluation, and also did not include them in the KP Corpus event and segment counts. Bryan Pardo’s original MIDI files for the KP Corpus also contain several missing chords, as well as chord labels that are shifted from their true onsets. We used chord and beat list files sent to us by David Temperley to correct these mistakes.

5.4 The Rock Dataset

To evaluate the system’s ability to recognize chords in a different genre, we compiled a corpus of 59 pop and rock songs from Hal Leonard’s The Best Rock Songs Ever (Easy Piano) songbook. Like the KP Corpus, the Rock dataset contains chords with added ninths—including major ninth chords and dominant seventh chords with a sharpened ninth—as well as inverted chords. We again omit the ninth and inversion numbers in these cases. Unique from the other datasets, the Rock dataset also possesses suspended and power chords. We extend our chord set to include these, adding suspended second, suspended fourth, dominant seventh suspended fourth, and power chords. We use the major mode canonical root set for suspended second and power chords and the minor canonical root set for suspended fourth chords, as this configuration produces the least number of accidentals. In all, there are 12 roots × 3 triad modes × 4 added notes + 12 roots × 4 sus and pow modes = 192 possible labels, with only 48 appearing in the dataset.

Similar to the KP Corpus, unlabeled segments occur at the beginning of some songs, which we omit from evaluation. Additionally, the Rock dataset uses an N.C. (i.e. no chord) label for some segments within songs where the chord is unclear. We broke songs containing this label into subsections consisting of the segments occurring before and after each N.C. segment, discarding subsections less than three measures long.

To create the Rock dataset, we converted printed sheet music to MusicXML files using the optical mu-
sic recognition (OMR) software PhotoScore. We noticed in the process of making the dataset that some of the originally annotated labels were incorrect. For instance, some segments with added note labels were missing the added note, while other segments were missing the root or were labeled with an incorrect mode. We automatically detected these cases and corrected each label by hand, considering context and genre-specific theory. We also omitted two songs ('Takin’ Care of Business’ and ‘I Love Rock N’ Roll’) from the 61 songs in the original Hal Leonard songbook, the former because of its atonality and the latter because of a high percentage of mistakes in the original labels.

6. Experimental Evaluation
We implemented the semi-Markov CRF chord recognition system using a multi-threaded package that has been previously used for noun-phrase chunking of informal text (Muis and Lu, 2016). The following sections describe the experimental results obtained on the four datasets from Section 5 for: our semi-CRF system; Radicioni and Esposito’s perceptron-trained HMM system, HMPerceptron; and Temperley’s computational music system, Melisma Music Analyzer.

6.1 BaCh Evaluation
We evaluated the semi-CRF model on BaCh using 10-fold cross validation: the 60 Bach Chorales were randomly split into a set of 10 folds, and each fold was used as test data, with the other nine folds being used for training. We then evaluated HMPerceptron using the same randomly generated folds to enable comparison with our system. However, we noticed that the performance of HMPerceptron could vary significantly between two different random partitions of the data into folds. Therefore, we randomly generated 10 fold sets, each fold set containing a different partition of the dataset into 10 folds. We then averaged the results of both systems over the 10 fold sets.

For our system, we computed the frequency of occurrence of each feature in the training data, using only the true segment boundaries and their labels. To speedup training and reduce overfitting, we only used features whose counts were at least 5. The performance measures were computed by averaging the results from the 10 test folds for each of the fold sets. Table 1 shows the averaged event-level and segment-level performance of the semi-CRF model, together with two versions of the HMPerceptron: HMPerceptron1, for which we do enharmonic normalization both on training and test data, similar to the normalization done for semi-CRF; and HMPerceptron2, which is the original system from (Radicioni and Esposito, 2010) that does enharmonic normalization only on test data. When computing the segment-level performance, a predicted segment is considered correct only if both its boundaries and its label match those of the true segment.

The semi-CRF model achieves a 6.2% improvement in event-level accuracy over the original HMPerceptron model, which corresponds to a 27% relative error reduction. The improvement in accuracy over HMPerceptron1 is statistically significant at an averaged p-value of 0.001, using a one-tailed Welch’s t-test over the sample of 60 chorale results for each of the 10 fold sets. The improvement in segment-level performance is even more substantial, with a 7.8% absolute improvement in F-measure over the original HMPerceptron model, and a 7.5% improvement in F-measure over the HMPerceptron1 version, which is statistically significant at an averaged p-value of 0.002, using a one-tailed Welch’s t-test. As HMPerceptron1 outperforms HMPerceptron2 in both event and segment-level accuracies, we will use HMPerceptron1 for the remaining evaluations and will simply refer to it as HMPerceptron.

<table>
<thead>
<tr>
<th>BaCh: Full chord evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
</tr>
<tr>
<td>semi-CRF</td>
</tr>
<tr>
<td>HMPerceptron1</td>
</tr>
<tr>
<td>HMPerceptron2</td>
</tr>
</tbody>
</table>

Table 1: Comparative results (%) on the BaCh dataset, using Event-level accuracy (AccE) and Segment-level precision (P), recall (RS), and F-measure (FS).

We also evaluated performance in terms of predicting the correct root of the chord, e.g. if the true chord label were Cmaj, a predicted chord of C7 would still be considered correct, because it has the same root as the correct label. We performed this evaluation for semi-CRF, HMPerceptron, and the harmony component of Temperley’s Melisma. Results show that semi-CRF improves upon the event-level accuracy of HMPerceptron by 5.1%, producing a relative error reduction of 33.8%, and that of Melisma by 3.9%. Semi-CRF also achieves an F-measure that is 8.0% higher than HMPerceptron and 7.3% higher than Melisma. These improvements are statistically significant with a p-value of 0.01 or less using a one-tailed Welch’s t-test.

<table>
<thead>
<tr>
<th>BaCh: Root only evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
</tr>
<tr>
<td>semi-CRF</td>
</tr>
<tr>
<td>HMPerceptron</td>
</tr>
<tr>
<td>Melisma</td>
</tr>
</tbody>
</table>

Table 2: Root only results (%) on the BaCh dataset, using Event-level accuracy (AccE) and Segment-level precision (P), recall (R), and F-measure (F).

For our system, we computed the frequency of occurrence of each feature in the training data, using only the true segment boundaries and their labels. To speedup training and reduce overfitting, we only used features whose counts were at least 5. The performance measures were computed by averaging the results from the 10 test folds for each of the fold sets. Table 1 shows the averaged event-level and segment-level performance of the semi-CRF model, together with two versions of the HMPerceptron: HMPerceptron1, for which we do enharmonic normalization both on training and test data, similar to the normalization done for semi-CRF; and HMPerceptron2, which is the original system from (Radicioni and Esposito, 2010) that does enharmonic normalization only on test data. When computing the segment-level performance, a predicted segment is considered correct only if both its boundaries and its label match those of the true segment.

The semi-CRF model achieves a 6.2% improvement
6.1.1 BaCh Error Analysis

Error analysis revealed wrong predictions being made on chords that contained dissonances that spanned the duration of the entire segment (e.g., a second above the root of the annotated chord), likely due to an insufficient number of such examples during training. Manual inspection also revealed a non-trivial number of cases in which we disagreed with the manually annotated chords, e.g., some chord labels were clear mistakes, as they did not contain any of the notes in the chord. This further illustrates the necessity of building music analysis datasets that are annotated by multiple experts, with adjudication steps akin to the ones followed by TAVERN.

6.2 TAVERN Evaluation

To evaluate on the TAVERN corpus, we created a fixed training-test split: 6 Beethoven sets (B063, B064, B065, B066, B068, B069) and 4 Mozart sets (K025, K179, K265, K353) were used for testing, while the remaining 11 Beethoven sets and 6 Mozart sets were used for training. All sets were normalized enharmonically before being used for training or testing. Table 3 shows the event-level and segment-level performance of the semi-CRF and HMPerceptron model on the TAVERN dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Acc_E</th>
<th>P_S</th>
<th>R_S</th>
<th>F_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>78.0</td>
<td>67.3</td>
<td>60.9</td>
<td>64.0</td>
</tr>
<tr>
<td>HMPerceptron</td>
<td>57.3</td>
<td>19.6</td>
<td>13.6</td>
<td>16.1</td>
</tr>
</tbody>
</table>

Table 3: Event (Acc_E) and Segment-level (P_S, R_S, F_S) results (%) on the TAVERN dataset.

As shown in Table 3, semi-CRF outperforms HMPerceptron by 20.7% for event-level chord evaluation and by 48.0% in terms of chord-level F-measure. Root only evaluations provided in Table 4 reveal that semi-CRF improves upon HMPerceptron’s event-level root accuracy by 20.1% and Melisma’s event accuracy by 10.0%. Semi-CRF also produces a segment-level F-measure value that is with 53.2% higher than that of HMPerceptron and with 19.0% higher than that of Melisma. These improvements are statistically significant with a p-value of 0.01 or less using a one-tailed Welch’s t-test.

<table>
<thead>
<tr>
<th>System</th>
<th>Acc_E</th>
<th>P_S</th>
<th>R_S</th>
<th>F_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>86.0</td>
<td>74.6</td>
<td>68.4</td>
<td>71.4</td>
</tr>
<tr>
<td>HMPerceptron</td>
<td>65.9</td>
<td>22.9</td>
<td>15.1</td>
<td>18.2</td>
</tr>
<tr>
<td>Melisma</td>
<td>76.0</td>
<td>52.5</td>
<td>52.3</td>
<td>52.4</td>
</tr>
</tbody>
</table>

Table 4: Event (Acc_E) and Segment-level (P_S, R_S, F_S) results (%) on the TAVERN dataset.

6.2.1 TAVERN Error Analysis

The results in Tables 1 and 3 show that chord recognition is substantially more difficult in the TAVERN dataset than in BaCh. The comparatively lower performance on TAVERN is likely due to the substantially larger number of figurations and higher rhythmic diversity of the variations compared to the easier, mostly note-for-note texture of the chorales. Error analysis on TAVERN revealed many segments where the first event did not contain the root of the chord. For such segments, HMPerceptron incorrectly assigned chord labels whose root matched the bass of this first event. Since a single wrongly labeled event invalidates the entire segment, this can explain the larger discrepancy between the event-level accuracy and the segment-level performance. In contrast, semi-CRF assigned the correct labels in these cases, likely due to its ability to exploit context through segment-level features, such as the chord root coverage feature f_i and its duration-weighted version f_i.

6.3 KP Corpus Evaluation

To evaluate on the full KP Corpus dataset, we split the songs into 11 folds. With this configuration, the first 9 folds each contain 4 songs, while the remaining 2 folds contain 5 songs. We then created two versions of semi-CRF: the original system with augmented 6th chord features omitted (semi-CRF_1) and an updated version with augmented 6th features added (semi-CRF_2). We tested all 46 songs on both versions, as shown in Table 5. We could not perform the same evaluation on HMPerceptron because it is not designed to handle augmented 6th chords.

<table>
<thead>
<tr>
<th>System</th>
<th>Acc_E</th>
<th>P_S</th>
<th>R_S</th>
<th>F_S</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF_1</td>
<td>72.0</td>
<td>59.0</td>
<td>49.2</td>
<td>53.5</td>
</tr>
<tr>
<td>semi-CRF_2</td>
<td>73.4</td>
<td>59.6</td>
<td>50.1</td>
<td>54.3</td>
</tr>
</tbody>
</table>

Table 5: Event (Acc_E) and Segment-level (P_S, R_S, F_S) results (%) on the KP Corpus dataset.

The results in Table 5 demonstrate the utility of adding augmented 6th chord features to our system, as semi-CRF_2 outperforms semi-CRF_1 on all measures. We will use semi-CRF_2 for the rest of the evaluations in this section, simply calling it semi-CRF.

We additionally perform root only evaluation on the full dataset for semi-CRF and Melisma. We ignore events that belong to the true augmented 6th chord segments when computing the root accuracies for both systems, as augmented 6th chords technically do not contain a root note. As shown in Table 6, Melisma is only marginally better than semi-CRF in terms of event-level root accuracy, however it has a segment-level F-measure that is with 2.5% better.

To enable comparison with HMPerceptron, we also evaluate all systems on the 36 songs that do not con-
We split the 59 songs in the rock dataset into 10 folds: 9 folds with 6 songs and 1 fold with 5 songs. Similar to the full KP Corpus evaluation from Section 6.3, we create two versions of the semi-CRF model. The first is the original semi-CRF system (semi-CRF₁) which does not contain suspended and power chord features. The second is a new version of semi-CRF (semi-CRF₂) which has suspended and power chord features added to it. We do not include HMPerceptron in the evaluation of the full dataset, as it is not designed for suspended and power chords.

As shown in Table 9, semi-CRF₂ obtains higher event and segment-level accuracies than semi-CRF₁. Therefore, we use semi-CRF₂ for the rest of the experiments, simply calling it semi-CRF.

We perform root only evaluation on the full Rock dataset using semi-CRF and Melisma. In this case, it is not necessary to omit the true segments whose labels are suspended or power chords, as these types of chords contain a root. As shown in Table 10, semi-CRF outperforms Melisma on all measures: it obtains a 6.9% improvement in event-level root accuracy and a 29.5% improvement in segment-level F-measure over Melisma.

We also evaluate only on the 51 songs that do not contain suspended or power chords to compare semi-CRF against HMPerceptron. We do this by splitting the reduced number of songs into 10 folds: 9 folds with 5 test songs and 46 training songs, and 1 fold with 6 test songs and 45 training songs. Results demonstrate that semi-CRF performs better than HMPerceptron: it achieves a 5.7% improvement in event-level chord accuracy and a 16.3% improvement in F-measure over HMPerceptron. Additionally, we evaluate the root-level performance of all systems on the 51 songs. Semi-CRF achieves better root-level accuracy than both systems: it obtains a 2.4% improvement in event-level root accuracy over HMPerceptron and a 3.4% improvement over Melisma. In terms of segment-level accuracy, it demonstrates a 17.5% improvement in F-measure over HMPerceptron and a 20.5% improvement over Melisma.

### Table 6: Event (Accₐ) and Segment-level (Pₛ, Rₛ, Fₛ) results (%) on the KP Corpus dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accₐ</th>
<th>Pₛ</th>
<th>Rₛ</th>
<th>Fₛ</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>80.7</td>
<td>64.8</td>
<td>55.2</td>
<td>59.4</td>
</tr>
<tr>
<td>Melisma</td>
<td>80.8</td>
<td>60.6</td>
<td>63.3</td>
<td>61.9</td>
</tr>
</tbody>
</table>

### Table 7: Event (Accₐ) and Segment-level (Pₛ, Rₛ, Fₛ) results (%) on the KP Corpus dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accₐ</th>
<th>Pₛ</th>
<th>Rₛ</th>
<th>Fₛ</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>73.0</td>
<td>55.6</td>
<td>50.7</td>
<td>53.0</td>
</tr>
<tr>
<td>HMPerceptron</td>
<td>72.9</td>
<td>48.2</td>
<td>43.6</td>
<td>45.4</td>
</tr>
</tbody>
</table>

### Table 8: Event (Accₐ) and Segment-level (Pₛ, Rₛ, Fₛ) results (%) on the KP Corpus dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accₐ</th>
<th>Pₛ</th>
<th>Rₛ</th>
<th>Fₛ</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>79.3</td>
<td>61.8</td>
<td>56.4</td>
<td>59.0</td>
</tr>
<tr>
<td>HMPerceptron</td>
<td>79.0</td>
<td>54.7</td>
<td>49.9</td>
<td>51.9</td>
</tr>
<tr>
<td>Melisma</td>
<td>81.9</td>
<td>60.7</td>
<td>63.7</td>
<td>62.2</td>
</tr>
</tbody>
</table>

### Table 9: Event (Accₐ) and Segment-level (Pₛ, Rₛ, Fₛ) results (%) on the Rock dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accₐ</th>
<th>Pₛ</th>
<th>Rₛ</th>
<th>Fₛ</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF₁</td>
<td>66.0</td>
<td>49.8</td>
<td>47.3</td>
<td>48.5</td>
</tr>
<tr>
<td>semi-CRF₂</td>
<td>69.4</td>
<td>60.7</td>
<td>53.3</td>
<td>56.8</td>
</tr>
</tbody>
</table>

### Table 10: Event (Accₐ) and Segment-level (Pₛ, Rₛ, Fₛ) results (%) on the Rock dataset.

<table>
<thead>
<tr>
<th>System</th>
<th>Accₐ</th>
<th>Pₛ</th>
<th>Rₛ</th>
<th>Fₛ</th>
</tr>
</thead>
<tbody>
<tr>
<td>semi-CRF</td>
<td>85.8</td>
<td>70.9</td>
<td>63.1</td>
<td>66.8</td>
</tr>
<tr>
<td>Melisma</td>
<td>78.9</td>
<td>31.5</td>
<td>45.7</td>
<td>37.3</td>
</tr>
</tbody>
</table>
These results are statistically significant with a $p$-value of 0.01 or less using a one-tailed Welch’s $t$-test.

<table>
<thead>
<tr>
<th>Rock 51 songs: Full chord evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
</tr>
<tr>
<td>semi-CRF</td>
</tr>
<tr>
<td>HMPerceptron</td>
</tr>
</tbody>
</table>

Table 11: Event (ACC $E$) and Segment-level (P$_S$, R$_S$, F$_S$) results (%) on the Rock dataset.

<table>
<thead>
<tr>
<th>Rock 51 songs: Root only evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
</tr>
<tr>
<td>semi-CRF</td>
</tr>
<tr>
<td>HMPerceptron</td>
</tr>
<tr>
<td>Melisma</td>
</tr>
</tbody>
</table>

Table 12: Event (ACC $E$) and Segment-level (P$_S$, R$_S$, F$_S$) results (%) on the Rock dataset.

6.4.1 Rock Error Analysis

As mentioned in Section 5.4, we automatically detected and manually corrected a number of mistakes that we found in the original chord annotations. In some instances, although the root of the provided chord label was missing from the corresponding segment, the label was in fact correct. Often times in these cases, the root appeared in the previous segment and thus was still perceptually salient to the listener, either because of its long duration or because it appeared in the last event of the previous segment. Other times, the same harmonic and melodic patterns were repeated throughout the piece, with the root appearing in the first few repetitions of these patterns, but disappearing later on. This was true for ‘Twist and Shout’ by the Beatles, in which the same I IV V7 progression of C major, F major, and G major 7 is repeated throughout the song, with the root C disappearing from C major segments by measure 11. Due to their inability to exploit larger scale patterns, neither system could predict the correct label for such segments.

We also found that three of the songs that we manually detected as having labels with incorrect modes (‘Great Balls of Fire,’ ‘Heartbreak Hotel,’ and ‘Shake, Rattle, and Roll’) were heavily influenced by blues. The three songs contain many major chord segments where the major third is purposefully swapped for a minor third to create a blues feel. We kept the labels as they were in these instances, but again both systems struggled to correctly predict the true label in these cases.

7. Related Work

Numerous approaches for computerized harmonic analysis have been proposed over the years, starting with the pioneering system of Winograd (1968), in which a systemic grammar was used to encode knowledge of harmony. Barthelemy and Bonardi (2001) and more recently Rizo et al. (2016) provide a good survey of previous work in harmonic analysis of symbolic music. Here, we focus on the three systems that inspired our work: Melisma (Temperley and Sleator, 1999), HarmAn (Pardo and Birmingham, 2002), and HMPerceptron (Radicioni and Esposito, 2010) (listed in chronological order). These systems, as well as our semi-CRF approach, incorporate knowledge of music theory through manually defined rules or features. For example, the “compatibility rule” used in Melisma is analogous to the chord coverage features used in the semi-CRF, the “positive evidence” score computed based on the six template classes in HarmAn, or the “Asserted-notes” features in HMPerceptron. Likewise, the segment purity features used in semi-CRF are analogous to the “negative evidence” scores from HarmAn, while the figuration heuristics used in semi-CRF can be seen as the counterpart of the “ornamental dissonance rule” used in Melisma. In these systems, each rule or feature is assigned an importance, or weight, in order to enable the calculation of an overall score for any candidate chord segmentation. Given a set of weights, optimization algorithms are used to determine the maximum scoring segmentation and labeling of the musical input. HMPerceptron uses the Viterbi algorithm (Rabiner, 1989) to find the optimal sequence of event labels, whereas semi-CRF uses a generalization of Viterbi (Sarawagi and Cohen, 2004) to find the joint most likely segmentation and labeling. The dynamic programming algorithm used in Melisma is actually an instantiation of the same general Viterbi algorithm – like HMPerceptron and semi-CRF it makes a first order Markov assumption and computes a similar lattice structure that enables a linear time complexity in the length of the input. HarmAn, on the other hand, uses the Relaxation algorithm (Cormen et al., 2009), whose original quadratic complexity is reduced to linear through a greedy approximation.

While the four systems are similar in terms of the musical knowledge they incorporate and the optimization algorithms or their time complexity, there are two important aspects that differentiate them:

1. Are the weights learned from the data, or pre-specified by an expert? HMPerceptron and semi-CRF train their parameters, whereas Melisma and HarmAn have parameters that are predefined manually.
2. Is chord recognition done as a joint segmentation and labeling of the input, or as a labeling of event sequences? HarmAn and semi-CRF are in the segment-based labeling category, whereas Melisma and HMPerceptron are event-based.

Learning the weights from the data is more feasible, more scalable, and, given a sufficient amount of training examples, much more likely to lead to optimal per-
formance. Furthermore, the segment-level classification has the advantage of enabling segment-level features that can be more informative than event-level analogues. The semi-CRF approach described in this paper is the first to take advantage of both learning the weights and performing a joint segmentation and labeling of the input.

8. Future Work

Manually engineering features for chord recognition is a cognitively demanding and time consuming process that requires music theoretical knowledge and that is not guaranteed to lead to optimal performance, especially when complex features are required. In future work we plan to investigate automatic feature extraction using recurrent neural networks (RNN). While RNNs can theoretically learn useful features from raw musical input, they are still event-level taggers, even when used in more sophisticated configurations, such as bi-directional deep LSTMs (Graves, 2012). We plan to use the Segmental RNNs of Kong et al. (2016), which combine the benefits of RNNs and semi-CRFs: bidirectional RNNs compute representations of candidate segments, whereas segment-label compatibility scores are integrated using a semi-Markov CRF. Learning the features entirely from scratch could require a larger number of training examples, which may not be feasible to obtain. An alternative is to combine RNN sequence models with explicit knowledge of music theory, as was done recently by Jaques et al. (2017) for the task of melody generation.

Music analysis tasks are mutually dependent on each other. Voice separation and chord recognition, for example, have interdependencies, such as figuration notes belonging to the same voice as their anchor notes. Temperley and Sleator (1999) note that harmonic analysis, in particular chord changes, can benefit meter modeling, whereas knowledge of meter is deemed crucial for chord recognition. This “serious chicken-and-egg problem” can be addressed by modeling the interdependent tasks together, for which probabilistic graphical models are a natural choice. Correspondingly, we plan to develop models that jointly solve multiple music analysis tasks, an approach that reflects more closely the way humans process music.

9. Conclusion

We presented a semi-Markov CRF model that approaches chord recognition as a joint segmentation and labeling task. Compared to event-level tagging approaches based on HMMs or linear CRFs, the segment-level approach has the advantage that it can accommodate features that consider all the notes in a candidate segment. This capability was shown to be especially useful for music with complex textures that diverge from the simpler note-for-note structures of the Bach chorales. The semi-CRF’s parameters are trained on music annotated with chord labels, a data-driven approach that is more feasible than specifying values by hand, especially when the number of rules or features is large. Empirical evaluations on three datasets of classical music and a newly created dataset of rock music show that the semi-CRF model performs substantially better than previous approaches when trained on a sufficient number of labeled examples and stays competitive when the training data is small.

Notes

1. Link to Music21: http://web.mit.edu/music21
2. Link to TAVERN: https://github.com/jcdevaney/TAVERN
4. Link to Kostka-Payne corpus: http://www.cs.northwestern.edu/~pardo/kpcorpus.zip
5. Link to PhotoScore: http://www.neuratron.com/photoscore.htm
7. Link to David Temperley’s Melisma Music Analyzer: http://www.link.cs.cmu.edu/melisma/

Acknowledgments

We would like to thank Patrick Gray for his help with pre-processing the TAVERN corpus. We greatly appreciate the help provided by Daniele P. Radicioni and Roberto Esposito and their willingness to share the original BaCh dataset and the HMPeXecptron implementation. We would like to thank Bryan Pardo for sharing the KP Corpus and David Temperley for providing us with an updated version that fixed some of the labeling errors in the original KP corpus. We would also like to thank David Chelberg for providing valuable comments on a previous version of the manuscript.

Kristen Masada’s work on this project was partly supported by an undergraduate research apprenticeship grant from the Honors Tutorial College at Ohio University.

References


