BREVE: an HMPerceptron-Based Chord Recognition System

Daniele P. Radicioni and Roberto Esposito

Abstract Tonal harmony analysis is a sophisticated task. It combines general knowledge with contextual cues, and it is concerned with faceted and evolving objects such as musical language, execution style and taste. We present BREVE, a system for performing a particular kind of harmony analysis, *chord recognition*: music is encoded as a sequence of sounding events and the system should assing the appropriate chord label to each event. The solution proposed to the problem relies on a conditional model, where domain knowledge is encoded in the form of Boolean features. BREVE exploits the recently proposed algorithm `CarpeDiem` to obtain significant computational gains in solving the optimization problem underlying the classification process. The implemented system has been validated on a corpus of chorales from J.S. Bach: we report and discuss the learnt weights, point out the committed errors, and elaborate on the correlation between errors and growth in the classification times in places where the music is less clearly asserted.

Key words: Chord Recognition; Machine Learning; Music Analysis.

1 Introduction

The musical domain has always exerted a strong fascination on researchers from diverse fields. In the last few years a wealth of research effort has been invested to analyze music, under an academic and industrial pressure [31]. Techniques of intelligent music search and analysis are crucial to devise systems for various purposes, such as for music identification, for deciding on music similarity, for music classifi-

Daniele P. Radicioni
Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149, Torino e-mail: radicion@di.unito.it

Roberto Esposito
Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149, Torino e-mail: esposito@di.unito.it

cation based on some set of descriptors, for algorithmic playlist generation, and for music summarization. Indeed, recent technological advances significantly enhanced the way automatic environments compose music [5], expressively perform it [12], accompany human musicians [23], and the way music is sold and bought through web stores [11].

Music *analysis* is a necessary step for composing, performing and –ultimately– understanding music, for both human beings and artificial environments (see, e.g., the works in [30] and [18]). Within the broader area of music analysis, we single out the task of *chord recognition*. This is a challenging problem for music students, who spend considerable amounts of time in learning tonal harmony, as well as for automatic systems. It is an interesting problem, and a necessary step towards performing a higher-level structural analysis that considers the main structural elements in music in their mutual interconnections. In Western tonal music at each time point of the musical flow (or *vertical*) one can determine which chord is sounding: chord recognition typically consists in indicating the fundamental note (or *root*) and the mode of the chord (Figure 1).

In our present approach the analysis task is cast to a supervised sequential learning (SSL) problem. From a methodological viewpoint, we transport to the musical domain the state-of-the-art machine learning *conditional models* paradigm, originally devised for the part-of-speech (POS) tagging problem [4]. A set of Boolean *features* has been designed in the attempt to encode the main cues used by human experts to analyze music.

One particular difficulty in dealing with sequential prediction is the computational complexity of inference algorithms. Intuitively, this problem can be cast to a path finding problem over a layered graph (described below) with vertices representing chord labels. In particular, given T layers and K label classes per layer, the graph representing the corresponding search space is a $T \times K$ graph. The problem of finding an optimum path over such graph is customarily solved by using the Viterbi algorithm, a dynamic programming algorithm having $\Theta(TK^2)$ time complexity [29]. Instead, to perform such decoding step our system relies on a recently developed decoding algorithm, `CarpeDiem`, that finds the *optimal* path in $O(TK\log(K))$ time in the best case, degrading to Viterbi complexity in the worst case [9].

We presently illustrate BREVE, a system for chord recognition that takes as input musical pieces encoded as MIDI files, extracts the corresponding sequence of music events, and computes the corresponding sequence of chord labels. Our approach puts together various insights from the fields of Machine Learning, Cognitive Science and Computer Music in an interdisciplinary fashion. The work is structured as follows. We start by formulating the problem of chord recognition (Section 2). Subsequently, we survey some related works on the problem of chord recognition (Section 3). In Section 4 we introduce the system BREVE: we illustrate how musical information is represented (Section 4.1), we introduce tonal harmony analysis as a sequential problem (Section 4.2), and motivate the adoption of a conditional model (Section 4.3). In particular we introduce the Boolean features framework (Section 4.4), and we illustrate the set of implemented features (Section 4.5). In Section 4.6 we briefly summarize the functioning of the `CarpeDiem` algorithm

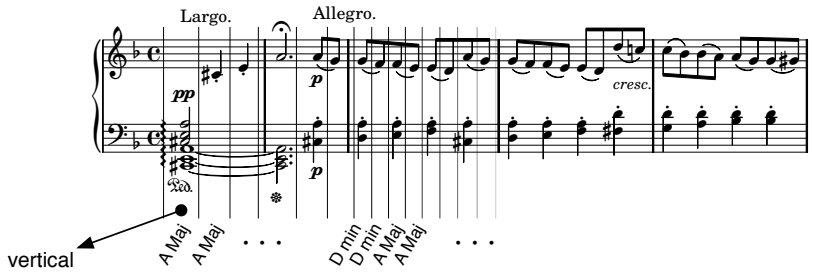


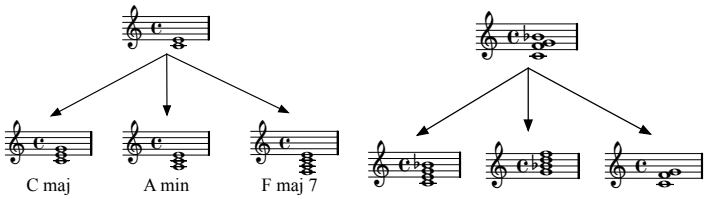
Fig. 1: The chord recognition problem consists of indicating for each vertical which chord is currently sounding. Excerpt from Beethoven's Piano Sonata Opus 31 n.2, 1st movement.

and then report the results of the experimentation in Section 5. We both examine in detail the computed weights to analyze which sorts of knowledge they overall describe (Section 5.1) and elaborate on the errors committed (Section 5.2). Finally, we draw some conclusions on future directions of BREVE and on the technologies currently adopted.

2 Chord Recognition Problem

The task of chord recognition consists in indicating a chord label for each music event in a music piece. Equivalently, one could individuate segments as portions of the piece with same harmonic content, and then assign the appropriate label to each segment [20, 27]. Several types of notation can be adopted in analysing music, such as Figured Bass, Roman numeral notation, classical letter, Jazz notation, and different representations for musical chord symbols are possible [13]. Furthermore, chord recognition is the first step toward a higher level structural analysis, mainly concerned with individuating the main building blocks of a composition along with the structural relationships underlying whole pieces [26].

We define the problem of chord recognition as follows. A *chord* is a set of (three or more) notes sounding at the same time. *Chord recognition* consists in indicating the fundamental note (or *root*) and the *mode* of the chord, e.g., *CMaj* or *Fmin*, at each time point of the musical flow (Figure 1). Given a score, we individuate sets of simultaneous notes (*verticals*), and associate to each vertical a *label* $\langle$*fundamental note*, *mode* $\rangle$. Additionally, we individuate the added notes that possibly enrich the basic harmony: we handle the cases of seventh, sixth and fourth. By considering 12 root notes $\times$ 3 possible modes (see below) $\times$ 3 possible added notes, we obtain 108 possible labels.



(a) Incompletely stated chords (b) Passing tones, retards

Fig. 2: (a) Cases in which triads are incompletely stated, and (b) in which triads are stated together with further notes like passing tones and retards can make chord recognition a harder problem.

Tonal harmony theory encodes two key aspects in music: how to build chords (that is, which simultaneous sounds are admissible)[1] and how to build successions of chords (that is which chord sequences are admissible). In the following we refer to them as *vertical* and *horizontal* information, respectively. For example, three main kinds or *modes* of chords are defined: *major*, *minor* and *diminished* chords. This sort of information is proper to states, and we denote it as *vertical* information. Moreover, tonal harmony theory encodes rules to concatenate chords, thus describing which successions are acceptable: for example, after a *CMaj* chord one could expect *FMaj* or *GMaj*, rather than $C\sharp Maj$. Typically, this sort of information can be represented as proper to transitions between states, and we denote it as *horizontal* information.

If we assign a label y to an event x and we only use information about the notes sounding around x, then our analysis relies on vertical information. By converse, if to predict the label y we consider only the previous label regardless the notes currently sounding, we are using only horizontal information.

Chord recognition is a hard task that requires integrating both kinds of information. In fact, music harmony can be incompletely stated (i.e., we are given only 2 elements of a chord, as in Figure 2(a)), or it can be stated by *arpeggio* (i.e., one note at a time); moreover, *passing tones*, *retards*, etc., can further complicate chord recognition (Figure 2(b)). Even fully stated chords can be ambiguous: let us consider, e.g., a chord composed of the notes *A-C-D-F*. This set of notes can be labeled as a *Dmin7* or *FMaj6*, depending on the inversion, on the harmonic flow of surrounding events, and on voicing information. In addition, one has to handle ambiguous cases, where the composer aims at violating the listener's expectation, deliberately contravening "grammatical" rules [3].

[1] Admissible are sounds perceived as *consonant* ones within a given musical style.

3 Related works

Much work has been carried out in the field of automatic tonal analysis: since the pioneering grammar-based work by Winograd [32], a number of approaches have been proposed that address the issue of tonal analysis. A survey of past works is provided by Barthelemy and Bonardi [1]; we focus on the closest approaches.

One of the preference-rules systems described by Temperley [28] is devoted to harmonic structure analysis. It relies on the Generative Theory of Tonal Music [17], providing that theory with a working implementation. In this approach, preference rules are used to evaluate possible analyses along a given dimension, such as harmony, also exploiting meter, grouping structure, and pitch spelling information. A major feature of Temperley's work concerns the application of high level domain knowledge, such as "Prefer roots that are close to the roots of nearby segments on the circle of fifths", which leads to an explanation of results.

The system by Pardo & Birmingham [21] is a simple template matching system. It performs tonal analysis by assigning a score to a set of 72 templates (that are obtained from the combination of 6 original templates transposed over 12 semitones of the chromatic scale). The resulting analysis is further improved by 3 tie-resolution rules, for labeling still ambiguous cases. This approach has been recently extended in the COCHONUT system by taking into account sequential information, in the form of chord sequence patterns [27].

Raphael & Stoddard [24] proposed a machine learning approach based on a Hidden Markov Model that computes Roman numeral analysis (that is, the higher level, functional analysis mentioned above). A main feature of their system is that the generative model can be trained using unlabeled data, thus determining its applicability also to unsupervised problems. In order to reduce the huge number of parameters to be estimated, they make a number of assumptions, such as that the current chord does not affect the key transitions. Also, the generative model assumes conditional independence of notes (the observable variables) given the current mode/chord.

Lee & Slaney [16] propose a system where 24 distinct HMMs are trained from acoustic signals. Two aspects are interesting in their work. First, by modelling 24 distinct HMMs they account for the differences in the chord transition probabilities that characterize different keys. Second, they train the HMMs using an input synthesized starting from a MIDI input. The MIDI files they use are downloaded from the Internet and annotated using Melisma music analyzer [28]. This approach has the advantage that large annotated corpora can be easily produced, however it can be argued that the final system learns the Melisma way of annotating music instead of the supposedly correct and unbiased way.

4 BREVE: an SSL System for Chord Recognition

In current Section we illustrate the design choices and the working of BREVE. We introduce the input representation, the Boolean-features framework, the motiva-

tions behind the sequential classification approach and the actual algorithms implemented.

4.1 Encoding

BREVE takes as input music pieces encoded as MIDI files. The fundamental representational structure processed is the music *event*; whole pieces are represented as *event lists*. An event is a set of *pitch classes* sounding at the same time; each new onset or offset determines a new event. Each event may be *accented* or *unaccented*. For each event we retain information about the bass.

Pitch classes are categories such that any two pitches one or more octaves apart are members of the same category. Their psychological reality received solid experimental evidence (e.g., see [6, 28]). Provided that we take as input MIDI files where pitch information is encoded as a number, pitch classes are computed by means of modulo-12 of MIDI pitches. E.g., if we consider the notes G3 and G4 corresponding to MIDI pitches 55 and 67, respectively, they both are mapped onto the same pitch class: $55 \equiv 67 \bmod 12$. Then, a vertical composed of the notes *C4-E4-G4* corresponding to the MIDI pitch numbers 60-64-67 is converted into an event composed of the pitch classes 0-4-7. Although loosing some information, pitch classes still permit to grasp the differences between chords. Also, they allow one to better characterize different modes. In fact, the characterizing aspect of a particular mode is the distance between the pitch classes that are present in the chord. For instance, distances intervening in $\langle 0,4,7 \rangle$, $\langle 0,3,7 \rangle$ and $\langle 0,3,6 \rangle$, correspond to major, minor and diminished modes respectively. In other terms, a chord mode is invariant under *rotations* of its constituting pitch classes (see Figure 3). If we consider major chords, whose pitches are 4 and then 3 semitones apart, we see that by rotating a major chord like *C Major* two 2 steps clockwise, we obtain the *D major* triad; by further rotating *D Major* 2 steps clockwise we obtain the *E Major* triad. The same holds if the chords are enriched with one or more added notes (Figure 3).

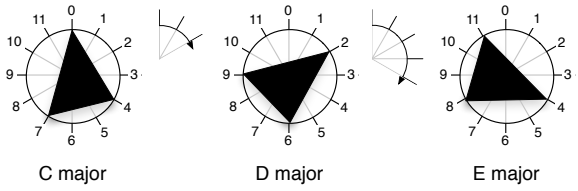


Fig. 3: The three main *modes* of chords.

In our representation, for each event we retain some information about the event *duration* as well: if a note *i* is held while a new note *j* is played, we consider an event containing *i*, and an event composed of both *i* and *j*, in that also the held note *i* affects

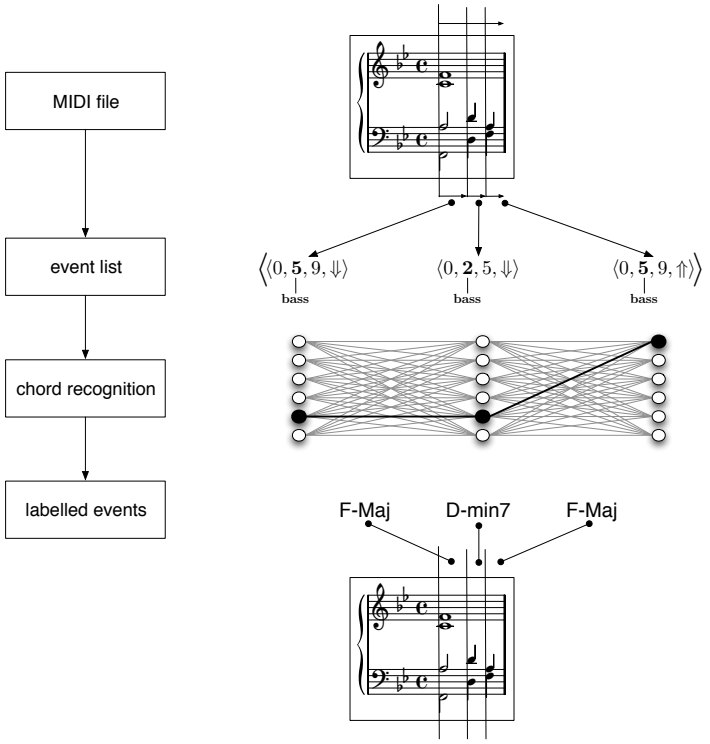


Fig. 4: The main steps performed by BREVE: it takes as input MIDI files, it extracts the event list with annotated bass and metrical accents (⇓ indicates an accented event, while ⇑ indicates an unaccented event) and then it assigns a chord label to each event.

the harmonic content of the vertical (see the $C4$ and $F4$ spanning over the second and third event in Figure 4). Most of the information required to perform tonal harmony analysis lies in the notes currently sounding and in their metrical salience: in particular, since harmonic content is mainly conveyed by accented events [17], we also annotate whether an event is accented or not. Meter estimation is based on the work of Temperley [28]. We annotate input events also with bass information, that provides valuable insights about inversions and more in general on harmonic flow. The final input representation and the information actually used through the analysis process is illustrated in Figure 4.

Our present representation disregards some musically relevant though secondary aspects, among which doubled notes, absolute pitches, pitch spelling, actual durations and voicing information are the most prominent.

4.2 Tonal Harmony Analysis as a Sequential Classification Problem

To analyze the harmonic structure of a piece is a sequential task, where contextual cues are widely acknowledged to play a fundamental role [2]. We have earlier argued that vertical and horizontal (that is, sequential) information grasp distinct though connected aspects of the musical flow. It follows that standard classification approaches such as decision trees, naive bayes, etc. are arguably less likely to produce good classification hypotheses, since they do not take into account the contextual information. This information is provided by surrounding observations, as well as nearby labeling.

In our view, the sequential aspects in the harmonic flow require to consider the problem of tonal harmony analysis as a sequential one. In particular, we adopt a Supervised Sequential Learning (SSL) approach. The task, known as the *supervised sequential learning* task, is not novel to the machine learning community. In recent years a wealth of research has been invested in developing algorithms to solve this kind of problem, and a number of interesting algorithms have been proposed [7, 4]. The SSL task can be specified as follows [8]:

Given: A set L of training examples of the form (X_m, Y_m), where each $X_m = (x_{m,1}, \ldots, x_{m,T_m})$ is a sequence of T_m feature vectors and each $Y_m = (y_{m,1}, \ldots, y_{m,T_m})$ is a corresponding sequence of class labels, $y \in \{1, ..., K\}$.

Find: A classifier H that, given a new sequence X of feature vectors, predicts the corresponding sequence of class labels $Y = H(X)$ accurately.

In our case, each X_m corresponds to a particular piece of music; $x_{m,t}$ is the information associated to the event at time t; and $y_{m,t}$ corresponds to the chord label (i.e., the chord root and mode) associated to the event sounding at time t. The problem is, thus, to learn how to predict accurately the chord label *given* the information on the music event.

4.3 Conditional Models Approach

The SSL problem can be solved with several techniques, such as Sliding Windows, Hidden Markov Models, Maximum Entropy Markov Models [19], Conditional Random Fields [15], and Collin's adaptation of the Perceptron algorithm to sequencial problems [4] (henceforth, HMPerceptron). All these methods, with the exception of Sliding Windows, are generalizations and improvements of Markovian sequence models, culminating in Conditional Random Fields and HMPerceptrons. Conditional Random Fields (CRFs) are state-of-the-art conditional probabilistic models, which improve on Maximum Entropy models [15, 7] while maintaining most of the beneficial properties of conditional models.

The major benefit of using conditional models is that fewer parameters need to be estimated at learning time. In fact, while generative models need to estimate the

"complete" distribution governing the random variables involved in the problem, conditional models only need to estimate the distribution of the output variables given the observed variables. This allows the learning algorithm to disregard many aspects of the problem that are not directly needed for the current classification task. On the other hand, in contrast with generative models which can be used to solve any conceivable inference problem, conditional models are specifically targeted to a given inference problem and cannot be extended beyond that.

CRFs, Maximum Entropy Markov Models and the HMPerceptron exploit the same way of interacting with the data. The "Boolean features" they use are functions of the current sequence. They are devised to return 1 if the label currently predicted for a variable of interest is coherent with the data, and to return 0 otherwise. Not only does this simplify the specification of the system, but also it allows domain knowledge to be easily plugged into the system. The algorithm, which is an extension to sequential problems of Rosenblatt's Perceptron algorithm [25], is reportedly at least on par with Maximum Entropy and CRFs models from the point of view of classification accuracy [4]. To the best of our knowledge, a direct comparison of HMPerceptron and CRFs has not been provided, even though they both were applied to the same Part-Of-Speech tagging problem, with analogous results [4, 15].

Therefore, on the basis of the above-mentioned literature, we have chosen the HMPerceptron as the main learning algorithm for the harmonic labeling prediction task. We briefly introduce the main facts about the working of the HMPerceptron, which has been investigated in [10].

The hypothesis acquired by the HMPerceptron has the form:

$$H(X) = \arg \max_{Y'=\{y'_1 \ldots y'_T\}} \sum_t \sum_s w_s \phi_s(X, y'_t, y'_{t-1})$$

where ϕ_s is a *Boolean feature*, i.e., it is a function of the sequence of events X and of the previous and current labels. The HMPerceptron has been defined within the Boolean features framework [19]. In this setting, the learnt classifier is built in terms of a linear combination of Boolean features. Each feature reports about a salient aspect of the sequence to be labelled in a given time instant. More formally, given a time point t, a Boolean feature is a $1/0$-valued function of the whole sequence of feature vectors X, and of a restricted neighborhood of y_t. The function is meant to return 1 if the characteristics of the sequence X around time step t support the classifications given at and around y_t.

4.4 The Boolean Features Framework and the HMPerceptron Algorithm

The ϕ_s functions are called *features* and allow the algorithm to take into consideration different aspects of the sequence being analyzed. To devise properly the set of features is a fundamental step of the system design, since this is the place where

both the domain knowledge and the model simplifications come into play. The w_s weights are the parameters that need to be estimated by the learning algorithm. To these ends, the HMPerceptron applies a simple scheme: it iterates over the training set updating the weights so that features correlated to correct outputs receive larger values and those correlated with incorrect ones receive smaller ones.

This is actually the same kind of strategy adopted by Rosenblatt's perceptron algorithm [25], the main difference between the two algorithms is in the way the hypothesis is evaluated. In the classification problem faced by the perceptron algorithm, in fact, it is sufficient to enumerate all the possible labels and to pick the best labelling. In the case of the SSL problem, however, this cannot be efficiently done: the labelling of a single "example" is a sequence of T labels, the number of such labelling is thus exponential in T. To enumerate all the possible label sequences and to pick the best is clearly unfeasible. In order to tame the complexity of the hypotheses evaluation, a first order Markov assumption can be made. In a conditioned model, this amounts to assume that the probability of observing any given label y_t depends only on the previous label y_{t-1}, given the observations: $\Pr(y_t|y_1,\ldots y_{t-1},X) = \Pr(y_t|y_{t-1},X)$.

The first order Markov assumption has two important consequences. The first one is that features cannot exploit any information about labels besides that currently assigned and the previous one. Vice versa, they can collect information from the whole sequence of observations. Secondly, the *Viterbi decoding* [22] can be used to decide about the best possible sequence of labels with a computational complexity in the order of $\Theta(TK^2)$. Viterbi can be interpreted as an optimum path finding algorithm, suitable for particular kinds of graphs. In our setting, the graph is constructed starting from $K \times T$ nodes: one for each label/time position pair (Figure 5). All vertices that correspond to a given time instant are fully connected to the nodes that correspond to the following time event (the absence of all other edges in the graph is the fingerprint of the first order Markov assumption). A left-to-right path in this graph corresponds to computing T labels, i.e., a labelling of T events. In this setting, the classifier acquired by the HMPerceptron can be though of as an assignment of weights to the vertices and the edges in the graph such that the best scoring path corresponds to the most likely sequence of labels.

The HMPerceptron algorithm estimates the weights associated to the features so to maximize H accuracy over the training set. The algorithm is very similar to Rosenblatt's Perceptron [25]: W is initialized to the zero vector; then, for each example (X_m, Y_m) in the training set, $H(X)$ is evaluated using the current W. Two situations may occur: the sequence of labels predicted by H is identical to Y_m or this is not the case, and a number of errors are committed. In the first case nothing is done for that example, and the algorithm simply jumps to the following one. In the second case the weight vector is updated using a rule very similar to the Perceptron update rule:

$$w_s = w_s + \sum_{t=1}^{T} \phi_s(X, y'_t, y'_{t-1})(I_{y'_t=y_t} - I_{y'_t \neq y_t}).$$

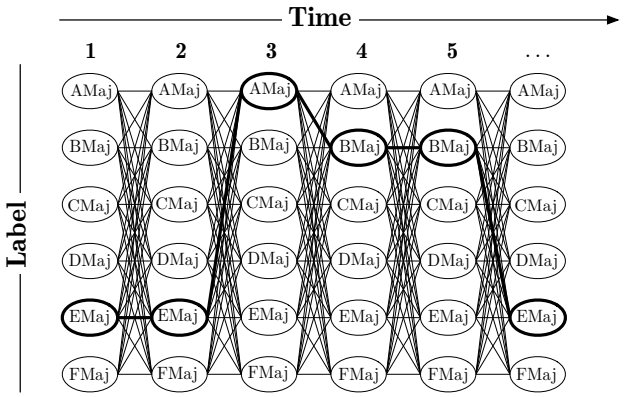


Fig. 5: Graphical representation of the labelling problem for $K = 6$.

In the formula ϕ_s represents a Boolean feature, y_t represents the t-th correct label, y'_t denotes currently predicted t-th label, and I_P represents the function that returns 1 in case P is verified and 0 otherwise. By noticing that (we omit the arguments of ϕ_s for brevity):

$$\phi_s \cdot (I_{y'_t=y_t} - I_{y'_t \neq y_t}) = \begin{cases} +1 & \text{if } \phi_s = 1 \wedge y'_t = y_t \\ -1 & \text{if } \phi_s = 1 \wedge y'_t \neq y_t \\ 0 & \text{if } \phi_s \text{ is } 0 \end{cases}$$

it is immediate to verify that the rule emphasizes features ϕ_s positively correlated with good classification and de-emphasizes those that are negatively correlated with it.

4.5 Features Design

The features are used to provide discriminative power to the learning system and incorporate domain knowledge. They are evaluated at each time point t in order to compute an informed prediction about the label to be assigned to the event. Formal definitions are reported in Table 1. Before illustrating the features in detail, let us start by summarizing few elements that have been introduced:

1. We distinguish between vertical and horizontal information. The former one is concerned with the portion of tonal harmonic theory that describes rules governing simultaneous sounds, whilst the latter one is concerned with the part of tonal harmony describing successions.

2. We cast the chord recognition problem to a supervised sequential learning problem, that can be solved as a path-finding problem by means of the HMPerceptron algorithm.
3. In particular, given T events and K labels available for each event, we build a T-layered graph. In this setting, labelling a sequence of chords corresponds to finding an optimum (with maximal reward) path throughout the graph, from the leftmost to the rightmost layer.

In the following, we describe the implemented set of features. We denote the current event with $x_{\cdot,t}$, and with y_t the currently predicted label.

Vertical features typically report about the presence (or absence) of some note in a given event, thus providing a proof for (or against) a given label. The features class Asserted-notes is composed of the features CompletelyStatedChord, AssertedAddedNote, ChordRootAssertedInTheNextEvent and AssertedRootNote. For instance, the feature AssertedRootNote reports about whether the root note of label y_t is present in event $x_{\cdot,t}$. This feature can provide precious cues, since the root note is the most salient sound in any chord. CompletelyStatedChord collects information about whether all the notes in chord y_t are present in event $x_{\cdot,t}$, and so forth.

Features named Asserted_v_NotesOfChord are triggered when *exactly* v notes of y_t are present in $x_{\cdot,t}$. In principle, the smaller is v, the lesser evidence exists for y_t. There are, thus, five different realizations of this feature. Some of them (for v= 3 or 4) strengthen current predictions, while those with low v are actually used to vote against the label y_t. In fact, cases where few notes of a predicted label are sounding should be taken as an evidence that the chord prediction is unreliable.

Features using bass information are grouped in the Bass-At-Degree features class, reporting information about which (if any) degree of the chord y_t is the bass of event $x_{\cdot,t}$. The bass is the most relevant pitch in any chord, often providing cues about the root note of current chord y_t. The other degrees of y_t can be present as the bass of the event $x_{\cdot,t}$. However this class of features should enable the system to apprehend that the third degree, the fifth degree and the possible added note are progressively less informative about the appropriate label.

Horizontal features have been arranged in two classes. One reports about how *meter* and *harmonic changes* relate. The other one reports about some transitions relevant in tonal harmony theory. ChordChanges features account for the correlation of label changes and the *beat level* of a neighborhood of $x_{\cdot,t}$. We distinguish two kinds of beat: accented and unaccented (1 and 0, respectively). In analyzing event $x_{\cdot,t}$, we consider also events $x_{\cdot,t-1}$ and $x_{\cdot,t+1}$. We denote with triplets of 1 and 0 the eight resulting metrical patterns. For instance, 010 denotes an accented event surrounded by two weak beats. The associated feature returns 1 if $y_{t-1} \neq y_t$ (i.e., the chord changes) in correspondence with such a pattern.

Human analysts are known to focus at first on a reduced set of transitions frequent in tonal music. Successions features are used to model them, as a way for biasing the system to behave accordingly. We represent a transition as a pattern, like $[Maj7, 5, Maj]$, composed of *i*) a starting mode and added note; *ii*) a distance between root notes, expressed as the number of intervening semitones; *iii*) an end-

Table 1: Boolean features formal definition. Definitions concerning a feature vector $x_{\cdot,\cdot}$ or concerning a label y are denoted by feature-name$[x_{\cdot,\cdot}]$ and by feature-name$[y]$, respectively. The intended meaning of each feature is reported in the *comment* column. Each definition reports the condition that is met when the feature returns 1. Also, each reported feature is actually a feature template which depends on several parameters. The parameters are mentioned in the feature name and explained in the *comment* column.

ϕ class	ϕ n.	ϕ name	Definition	Comments
Asserted-Notes	1	AssertedRootNote	$\text{root}[y_t] \in \text{notes}[x_{\cdot,t}]$	$\text{root}[y_t]$: root note of chord y_t
	2	ChordRootAssertedInTheNextEvent	$\text{root}[y_t] \in \text{notes}[x_{\cdot,t+1}]$	
	3	AssertedAddedNote	$\text{added}[y_t] \in \text{notes}[x_{\cdot,t}]$	$\text{added}[y_t]$: added note of chord y_t
	4	CompletelyStatedChord	$\forall i \in \{1..4\} : \text{note}_i[y_t] \in \text{notes}[x_{\cdot,t}]$	$\text{note}_i[y_t]$: i-th note in chord y_t
	5-9	Asserted_v_NotesOfChord	$\lvert\text{notes}[x_{\cdot,t}] \cap \text{notes}[y_t]\rvert = \mathrm{v}$	$\mathrm{v} \in \{0..4\}$; $\text{notes}[y_t]$: set of notes in y_t
Bass-At-Degree	10	BassIsRootNote	$\text{bass}[x_{\cdot,t}] = \text{root}[y_t]$	$\text{bass}[x_{\cdot,t}]$: bass of event $x_{\cdot,t}$
	11	BassIsThirdDegree	$\text{bass}[x_{\cdot,t}] = \text{third}[y_t]$	$\text{third}[y_t]$: third degree of chord y_t
	12	BassIsFifthDegree	$\text{bass}[x_{\cdot,t}] = \text{fifth}[y_t]$	$\text{fifth}[y_t]$: fifth degree of chord y_t
	13	BassIsAddedNote	$\text{bass}[x_{\cdot,t}] = \text{added}[y_t]$	
ChordChanges	14-21	ChordChangeOnMetricalPattern $\_i_1 i_2 i_3$	$\text{m}[x_{\cdot,t-1}] = i_1 \wedge$ $\text{m}[x_{\cdot,t}] = i_2 \wedge$ $\text{m}[x_{\cdot,t+1}] = i_3 \wedge$ $y_{t-1} \neq y_t$	$i_1, i_2, i_3 \in 0, 1$; $\text{m}[x_{\cdot,\cdot}]$ is 1 if event $x_{\cdot,\cdot}$ is accented and 0 otherwise
Successions	22-43	ChordDistance_m_1_i_m_2	$\text{mode}[y_{t-1}] = m_1 \wedge \text{mode}[y_t] = m_2 \wedge \text{root}[y_{t-1}] + i \equiv \text{root}[y_t] \pmod{12}$	$\text{root}[y_t]$: pitch class of root note of chord y_t

ing mode and added note. That is, $[Maj7, 5, Maj]$ refers to a transition from a major chord with added seventh to a major chord, whose roots are five semitones apart (e.g., from *FMaj7* to *B♭Maj*).

Both ChordChanges and Successions features can be used to capture musical aspects that are deeply ingrained with musical style. On the one hand, this implies that restrictive constraints are posed to the stylistic homogeneity between training and testing sets. On the other hand, such style sensitivity grasps valuable 'linguistic' aspects that characterize musical language.

It is relevant to point out that both vertical and horizontal features have been devised so as to *generalize* to unseen labels. For instance, CompletelyStatedChord will fire any time when all the notes of label y_t are present in $x_{\cdot,t}$, regardless of whether y_t was present in the training set or not. Similarly, the transition $[Maj7, 5, Maj]$ will provide useful information also about transitions between labels never met in the training set. This is in stark contrast with common learning systems. For instance, the transition matrix used in most HMMs-based systems provides detailed accounts of the probabilities of transitions which were observed in the training set. When used on a new dataset containing unseen transitions, the transition matrix cannot provide any useful information about it. On the contrary, based on the generalization power of the features, BREVE can abstract the transition between chords a perfect fifth apart, thereby recognizing transitions between any two chords such as (*FMaj7-B♭Maj*), (*C♯Maj7-F♯Maj*), (*D♭Maj7-G♭Maj*) as instances of a transition of type $[Maj7, 5, Maj]$.

To give an intuition of the generalization power of BREVE, we ran a preliminary experimentation. BREVE has been trained on two chorales by J.S. Bach: one in major key, one in minor key. It then has been tested on 58 chorales by the same author. We repeated this test 5 times, each time with two different training sequences. We obtained an average accuracy of 75.55%. This datum is surprisingly good if we consider that the whole dataset contains 102 different labels, while the training sets, each one composed of a pair of sequences only, contained 24.8 labels on average.

4.6 `CarpeDiem` algorithm

As mentioned, BREVE mainly exploits vertical information, resorting to horizontal cues primarily to resolve ambiguities. In other words, to label a given vertical, we *first* look at information provided by the current event, and *then* use surrounding context to make a decision if still in doubt. In BREVE, this 'least effort principle' is implemented by the `CarpeDiem` algorithm [9]. `CarpeDiem` is an algorithm allowing one to evaluate the best possible sequence of labels given the vertical and horizontal evidence. It solves the same problem solved by the Viterbi algorithm, but saves some computational effort. In particular, by exploiting vertical information `CarpeDiem` is able to reduce the number of labels taken into consideration by the system.

To provide an intuitive description of the algorithm, it is worth recalling that Viterbi algorithm [29] spends most computational resources to compute:

$$\max_{y_t, y_{t-1}} \left[\overbrace{\omega_{y_{t-1}}}^{\text{weight of the best path to } y_{t-1}} + \overbrace{\sum_s w_s \phi_s(X, y_t, y_{t-1}, t)}^{\text{weight for transition } y_{t-1}, y_t} \right]. \quad (1)$$

If we partition the set $\{1, 2, \ldots p\}$ of all feature *indexes* into the two sets Φ^0 and Φ^1, corresponding to indexes of vertical and horizontal features respectively, then the equation (1) can be rewritten into:

$$\max_{y_t} \left[\sum_{s \in \Phi^0} w_s \phi_s(X, y_t, t) + \max_{y_{t-1}} \left[\omega_{y_{t-1}} + \sum_{s \in \Phi^1} w_s \phi_s(X, y_t, y_{t-1}, t) \right] \right]. \quad (2)$$

Equation (2) is equivalent to Equation (1); in addition, it emphasizes how features in Φ^0 need to be evaluated only once per y_t label, and not once for each y_t,y_{t-1} pair. It is then obvious (and in accord with the intuition) that whenever $\Phi^1 = \emptyset$ the cost of the Viterbi algorithm can be reduced to be linear in the number of labels. The core idea underlying `CarpeDiem` is to exploit vertical information to avoid the evaluation of the inner maximization as long as possible.

5 Evaluation of the System

A corpus of 60 four-parts chorales harmonized by J.S. Bach (1675-1750) was annotated by a human expert with the bass and the appropriate chord label, as described in Section 4.1. Additional information on metrical salience of the music events has been computed by using the *meter* program by Temperley [28]. The sequences in the dataset are composed, on average, by 94.5 events; on the whole, the corpus contains 5,664 events.

BREVE has been validated using 10-fold cross validation: the estimated generalization error is 19.94%, thus providing an accuracy rate of 80.06%. Previous results of a preceding implementation of the system, tested on a less homogeneous dataset (namely, the Kostka-Payne corpus [14]), obtained a 73.8% accuracy. Also, BREVE exploits the state-of-the-art 'decoding' algorithm `CarpeDiem`, which allows BREVE to run in just 7.35% of the time required by an implementation based on the Viterbi algorithm [10].

In the following we elaborate on the *quality* of the output of BREVE and, specifically, *i*) we examine the learnt parameters from a musical perspective; *ii*) we inspect the errors committed in the classification and consider whether the analysis of errors can be useful in suggesting any improvement to the set of features or to the classification strategy.

In the next sections we elaborate on the weights learnt by the system and on the errors it commits. Since it is unclear how to aggregate the weights of the ten classi-

fiers acquired during the cross validation, we will consider the weights obtained by training an individual classifier on half of the music sequences in the dataset. Also, we will examine errors committed by the same classifier, which has been tested on the second half of the dataset.

5.1 Musical Interpretation of Acquired Classifiers

In Appendix 7 we present the features list with the learnt weights, arranged into the four classes outlined in Section 4.5.

As expected, information about which notes are currently sounding prevails over contextual information. The highest positive weights involve vertical features, namely Asserted_v_NotesOfChord with v=3 and v=2 (features number 7 and 8 in Table 2) and CompletelyStatedChord. In considering Asserted_v_NotesOfChord features, we see that the case v=3 received more emphasis with respect to the case v=4. This is likely due to the fact that in the examined corpus, events composed of 4 or more pitches mostly contain passing tones which mislead the Asserted_4_NotesOfChord feature. Furthermore, the lower frequency with which the feature is asserted may have affected the magnitude of the learned weight as well. As regards v=1, the feature has been used by the system as evidence *against* a given label, rather than supporting it. We also observe how the feature with v=1 received a penalty analogous, in magnitude, to the reward obtained by the feature with v=2. By looking at the AssertedDegrees features class, we observe that the features CompletelyStatedChord (feature number 4), AssertedRootNote (feature number 1), and Asserted_(2|3)_NotesOfChord necessarily fire simultaneously. In such situations, the 'amount of evidence' in favour of y_t is overwhelming and BREVE will probably investigate only few alternative labels.

Let us now consider horizontal features (numbered from 14 to 43). Horizontal features are arranged in two classes, Successions and ChordChanges. The former ones collect information about some recurrent transitions, whilst the latter ones are mainly concerned with the correlation between harmonic change and meter strength. Weights associated to harmony changes in correspondence with weak beats (patterns with shape ?0?) receive the highest penalty among all features. While harmony change is globally discouraged, the system assigns only weak penalties to harmonic changes in correspondence with accented beats (feature 17 over metrical pattern 011 and feature 20, over metrical pattern 110). The preference for chord transitions in correspondence with accented events substantially fits to experimental evidences and analytical strategies known in literature, e.g., the *Strong beat rule* proposed by Lerdahl & Jackendoff [17], and implemented by Temperley as the preference for "chord spans that start on strong beats of the meter" [28]. The speed of harmony changes is deeply with musical style as well as with the musical form under consideration: since these features are a simple –though effective– way of modeling the harmonic rhythm, they provide interesting high-level information about style.

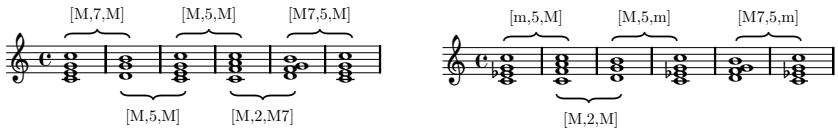


Fig. 6: The fundamental steps involved in cadence.

ChordDistance_m_1_i_m_2 features identify roughly three kinds of transitions: 1) those providing strong positive evidence in favor of a given label y_t; 2) those used as refinement criteria; 3) those used to forbid the transitions that have associated large negative numbers. All transitions involved in the *cadence* have been identified among the most relevant horizontal features (see features number 30, 22, 34, 25, 26, 32, 23, 24 in Table 2) and Figure 6.

Having identified the main components of the cadence shows that the system recognizes the relevance of such transitions in the considered corpus. In facts, four-parts chorales where harmonized by starting from melodies (properly, the chorales themselves) composed of phrases. Typically, the end of each phrase is marked by a cadence, so that the harmonization of chorales corresponds to a good extent to harmonizing the cadences, and then to connecting such 'fixed' points with further chords.

5.2 Errors Analysis

As mentioned, BREVE incorrectly classifies chords in about 20% of cases. A closer look at the errors reveals some well-defined error classes. First, in about 30% of errors, labels computed are wrong: in these cases BREVE has been totally misled. Also, determining why the system provided incorrect output is not simple. Somewhere, it has been caught in interpretative nuances that are hardly avoided given the simplified input representation (e.g., the system has no information on double notes, contrapuntal structure, which provide useful cues to human analysts). In other cases, the resistance to chord changes forces the system to inherit by mistake previously attributed labels (this is especially true for chord changes on weak beats).

Among other mistakes, only few errors appear execrable from the viewpoint of harmony theory. For instance, those due to confusing chord modes such as major with minor chords or viceversa. This is an error seldom committed by human analysts, and by BREVE, too. Precisely, only 8.60% (i.e., 1.71% of the labels in the dataset) of the overall errors fall in this class. One may argue that a lager context would be helpful to improve on this issue.

The errors due to confusion between relative keys (e.g., *Amin* instead of *Cmaj*) amount to 14.81% of the total error. Relative keys are intrinsically connected tones,

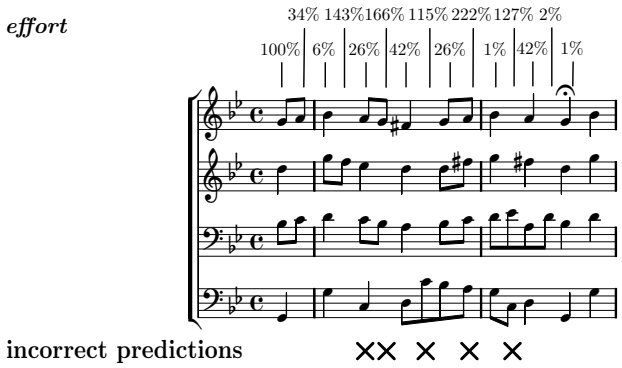


Fig. 7: How computational effort, in terms of label evaluations, and incorrect predictions co-occur. Excerpt taken from the four-parts chorale *BWV 6.6* by J.S. Bach.

in that they share the same *key signature* while having different roots, and to confuse between them is not a serious error from a musical viewpoint.

Many errors (namely 21.02%) are due to misclassifying the added note (root and mode being correct): in these cases the chord is analyzed in a substantially correct fashion. In most cases such errors are to be imputed to the resistance of the system to chord changes. For instance, this is the case of the succession *FMaj* $\Downarrow$-*FMaj* $\Uparrow$-*B*$\flat$*Maj* $\Downarrow$ when the seventh occurs in correspondence with the second –*unaccented*– *FMaj* chord. This case can be controversial even for human analysts, in that both *FMaj* and *FMaj*7 can be considered correct classifications for the second vertical. A human expert would make a decision inspecting whether the *E*$\flat$ can be considered a passing note or not. For example, a clue in favor of *FMaj*7 would be provided by the resolution of the seventh a semitone below, on the third degree of the new chord.

Additionally, we observe that many errors occur on unaccented beats. Namely, if we consider that only 34.87% of events in the dataset fall on unaccented beats, the fact that 38.85% of errors involve weak events shows that some improvement in this direction is possible.

On average, BREVE inspects only a fraction of all possible labels.[2] If we consider that the complexity of the Viterbi is $\Theta(TK^2)$ (see Figure 5), this datum help explaining how the implementation of BREVE based on `CarpeDiem` saves 92.65% of computation time with respect to implementation based on the Viterbi algorithm.

In particular, we can define an *effort measure* to characterize the amount of problem space explored, and compute such *effort* as the percentage of labels inspected by `CarpeDiem` with respect to Viterbi to compute the optimal label for current event. Such figure surpasses 100% when `CarpeDiem` visits some labels in previous layers (of course, this happens if the corresponding vertices had been left

[2] We note that the number of inspected nodes reported here is not a direct measure of the time required by the algorithm to run over the dataset. In facts, even for nodes that we consider as visited, not necessarily all incoming edges have been inspected by `CarpeDiem`.

previously unvisited). Since on average `CarpeDiem` does not explore more than 100% of available labels, this fact provides an intuitive argument to corroborate the fact that `CarpeDiem` is never asymptotically worse than the Viterbi algorithm [9]. Interestingly, we observed that such *effort* correlates (though only mildly) with the classification error. On average, in correspondence with correctly predicted labels BREVE inspected the 65.47% of available labels; on the contrary, in correspondence with incorrectly predicted labels, BREVE inspected the 78.59% of available labels. In Figure 7 we report an excerpt with annotated above the staves the *effort* required by the analysis, along with –below the staves– the events for which BREVE provided incorrect predictions. As outlined above, errors are more frequent in correspondence with weak beats, where harmony quickly changes. For example, the first event in measure 3 only requires considering the 1% of available labels, whilst the subsequent, unaccented, event requires inspecting 127% of labels (thus implying the need to backtrack to previous level of the graph). This is clearly due to a harmonic movement faster than usual, where evidence provided by vertical features (considering the asserted pitches) is contradicted by the horizontal ones, highly penalizing label changes on unaccented events surrounded by accented events (see feature number 19 in Table 2).

In summary, our review of the errors committed reveals that most errors are either venial, or justifiable on musical accounts. Moreover, more difficult passages require increased effort, and BREVE has been proved to spend more computational resources exactly on those events.

6 Conclusions

This chapter described BREVE, a system for chord recognition, a task that is at the ridge of AI, Cognitive Science and Computer Music. The approach implemented by BREVE is to cast the chord recognition problem to a Supervised Sequential Learning approach: the musical flow is mapped onto a sequence of events, each one labelled with a chord. BREVE exploits a corpus of annotated musical pieces in order to 'learn' how to label new excerpts.

Among several possible approaches to solve this kind of learning problems, BREVE exploits the HMPerceptron algorithm: it converges faster than most competitor approaches, meanwhile retaining comparable classification performance. In BREVE few tens of Boolean features encode rich domain knowledge. This is in stark contrast with many recently proposed generative models where the number of parameters to be learnt ranges in the thousands.

The experiments over a corpus of chorales from J.S. Bach show that the system has performances similar to competitor systems. However, the fewer number of parameters involved in BREVE arguably allows for better explanation of the results and a deeper understanding of the system. In facts, the learnt weights turned out to be be musically meaningful; also the errors committed fall in few and clearly identifiable classes, therefore leaving room to future refinements.

Future refinements may address two aspects: from a musical viewpoint, the music representation could be extended so as to allow taking into account the role played by key as a center of gravity of the whole composition, and functional analysis. From a machine learning perspective, the system could be extended to take into account also information about transitions between non-adjacent layers, thereby exploiting higher order Markov assumptions.

Acknowledgements

This research has been partly supported by the postdoctoral research grant Università di Torino – A.A, 200.102 POSTDOC.

References

1. Barthelemy, J., Bonardi, A.: Figured bass and tonality recognition. In: Procs. of the 2^{nd} Annual International Symposium on Music Information Retrieval 2001 (2001)
2. Bent, I. (ed.): Music Analysis in the Nineteenth Century. Cambridge University Press, Cambridge (1994)
3. Bigand, E., Pineau, M.: Global context effects on musical expectancy. Perception and psychophysics **59**(7), 1098–1107 (1997)
4. Collins, M.: Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2002). URL `http://citeseer.ist.psu.edu/collins02discriminative.html`
5. Cope, D.: A Musical Learning Algorithm. Comput. Music J. **28**(3), 12–27 (2004)
6. Demany, L., Semal, C.: Harmonic and melodic octave templates. The Journal of the Acoustical Society of America **88**(5), 2126–2135 (1990)
7. Dietterich, T.: Structural, Syntactic, and Statistical Pattern Recognition, *Lecture Notes in Computer Science*, vol. 2396, chap. Machine Learning for Sequential Data: A Review, pp. 15–30. Springer-Verlag (2002)
8. Dietterich, T.G., Ashenfelter, A., Bulatov, Y.: Training conditional random fields via gradient tree boosting. In: ICML '04: Twenty-first international conference on Machine learning. ACM Press, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1015330.1015428
9. Esposito, R., Radicioni, D.P.: `CarpeDiem`: an Algorithm for the Fast Evaluation of SSL Classifiers. In: Proceedings of the 24th Annual International Conference on Machine Learning (ICML 2007) (2007)
10. Esposito, R., Radicioni, D.P.: Trip Around the HMPerceptron Algorithm: Empirical Findings and Theoretical Tenets. In: M.T. Pazienza, R. Basili (eds.) AI*IA 2007: Advances in Artificial Intelligence, 10th Congress of the Italian Association for Artificial Intelligence. Springer-Verlag (2007)
11. Freeman, J.: Fast generation of audio signatures to describe iTunes libraries. Journal of New Music Research **35**(1), 51–61 (2006)
12. Grachten, M., Arcos, J.L., López de Mantaras, R.: A Case Based Approach to Expressivity-Aware Tempo Transformation. Mach. Learn. **65**(2–3), 411–437 (2006)
13. Harte, C., Sandler, M., Abdallah, S., Gomez, E.: Symbolic Representation of Musical Chords: A Proposed Syntax for Text Annotation. In: International Conference on Music Information Retrieval, pp. 66–71. Queen Mary, University of London (2005)

14. Kostka, S., Payne, D.: Tonal Harmony. McGraw-Hill, New York, NY (1984)
15. Lafferty, J., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Procs. of the 18th International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco, CA (2001)
16. Lee, K., Slaney, M.: Acoustic Chord Transcription and Key Extraction from Audio Using Key-Dependent HMMs Trained on Synthesized Audio. In: IEEE Transactions on Audio, Speech and Language Processing, vol. 16, pp. 291–301 (2008)
17. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge, MA (1983)
18. López de Mantaras, R., Arcos, J.: AI and Music: From Composition to Expressive Performance. AI Magazine **23**, 43–57 (2002)
19. McCallum, A., Freitag, D., Pereira, F.: Maximum entropy Markov models for information extraction and segmentation. In: Proc. 17th International Conf. on Machine Learning, pp. 591–598. Morgan Kaufmann, San Francisco, CA (2000). URL `citeseer.ist.psu.edu/mccallum00maximum.html`
20. Pardo, B., Birmingham, W.P.: Automated partitioning of tonal music. In: Procs. of the Thirteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS-2000 (2000)
21. Pardo, B., Birmingham, W.P.: Algorithms for chordal analysis. Comput. Music J. **26**, 27–49 (2002)
22. Rabiner, L.R.: A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: Proceedings of the IEEE, vol. 77, pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1989)
23. Raphael, C.: A Hybrid Graphical Model for Rhytmic Parsing. Artif. Intell. **137**(1-2), 217–238 (2002)
24. Raphael, C., Stoddard, J.: Functional harmonic analysis using probabilistic models. Computer Music Journal **28**(3), 45–52 (2004)
25. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review (Reprinted in Neurocomputing (MIT Press, 1998)) **65**, 386–408 (1958)
26. Schoenberg, A.: Structural Functions in Harmony. Norton, New York (1969)
27. Scholz, R., Ramalho, G.: Cochonut: Recognizing Complex Chords from MIDI Guitar Sequences. In: Procs. of the 2008 International Conference on Music Information Retrieval, pp. 27–32 (2008)
28. Temperley, D.: The Cognition of Basic Musical Structures. MIT Press, Cambridge, MASS (2001)
29. Viterbi, A.J.: Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. In: IEEE Transactions on Information Theory, vol. 13, pp. 260–269 (1967)
30. Widmer, G.: Discovering Simple Rules in Complex Data: A Meta-learning Algorithm and Some Surprising Musical Discoveries. Artif. Intell. **146**, 129–148 (2001)
31. Widmer, G.: Guest editorial: Machine learning in and for music. Machine Learning **65**, 343–346 (2006)
32. Winograd, T.: Linguistics and computer analysis of tonal harmony. Journal of Music Theory **12**, 2–49 (1968)

7 Features used by BREVE

Table 2: The feature list with the learnt weights. For each feature ϕ we report its class, its number, its name, and the weight learnt by the HMPerceptron.

ϕ class	ϕ number	ϕ name	weight
Asserted-notes	1	AssertedRootNote	2.600
	2	ChordRootAssertedInTheNextEvent	12.000
	3	AssertedAddedNote	-3.200
	4	CompletelyStatedChord	45.400
	5	Asserted_0_NotesOfChord	-70.600
	6	Asserted_1_NotesOfChord	-23.400
	7	Asserted_2_NotesOfChord	22.800
	8	Asserted_3_NotesOfChord	52.000
	9	Asserted_4_NotesOfChord	19.200
Bass-At-Degree	10	BassIsRootNote	17.400
	11	BassIsThirdDegree	13.400
	12	BassIsFifthDegree	0.200
	13	BassIsAddedNote	-0.600
ChordChanges	14	ChordChangeOnMetricalPattern_000	-9.600
	15	ChordChangeOnMetricalPattern_001	-95.200
	16	ChordChangeOnMetricalPattern_010	-23.600
	17	ChordChangeOnMetricalPattern_011	-1.800
	18	ChordChangeOnMetricalPattern_100	-84.600
	19	ChordChangeOnMetricalPattern_101	-552.200
	20	ChordChangeOnMetricalPattern_110	-10.800
	21	ChordChangeOnMetricalPattern_111	-82.200
Successions	22	ChordDistance_M_5_M	20.000
	23	ChordDistance_M_5_m	8.200
	24	ChordDistance_M7_5_m	21.600
	25	ChordDistance_M7_5_M	18.600
	26	ChordDistance_m_5_M	4.000
	27	ChordDistance_m_5_M7	-2.400
	28	ChordDistance_m7_5_M	6.800
	29	ChordDistance_m7_5_M7	0.400
	30	ChordDistance_M_7_M	13.800
	31	ChordDistance_M_7_M7	-13.200
	32	ChordDistance_M_2_M	8.200
	33	ChordDistance_M_2_m	7.800
	34	ChordDistance_M_2_M7	18.800
	35	ChordDistance_m6_2_M	-3.600
	36	ChordDistance_m6_2_M7	1.200
	37	ChordDistance_d_1_M	6.800
	38	ChordDistance_d_1_m	1.400
	39	ChordDistance_m_3_M	-36.600
	40	ChordDistance_m_8_M	-17.400
	41	ChordDistance_M_9_m	15.600
	42	ChordDistance_M_9_m7	-13.000
	43	ChordDistance_M4_0_M7	0.400