Proceedings of 20th International Symposium on Music Acoustics
(Associated Meeting of the International Congress on Acoustics)

25-31 August 2010, Sydney and Katoomba, Australia

Unsupervised Incremental Learning and Prediction of Audio Signals

Ricard Marxer and Hendrik Purwins

Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain

PACS: 43.75.St 43.75.Xz 43.75.Zz 43.75.Cd

ABSTRACT

The artful play with the listener's expectations is one of the supreme skills of a gifted musician. We present a system that analyzes an audio signal in an unsupervised manner in order to generate a musical representation of it on-the-fly. The system performs the task of next note prediction using the emerged representation. The main difference between our system and other existing music prediction systems is the fact that it dynamically creates the necessary representations as needed. Therefore it can adapt itself to any type of sounds, with as many timbre classes as there may be. The system consists of a conceptual clustering algorithm coupled with a modified hierarchical N-gram. The main flow of the system can be summarized in the following processing steps: 1) segmentation by transient detection, 2) timbre representation of each segment by Mel-cepstrum coefficients, 3) discretization by conceptual clustering, yielding a number of different sound classes (e.g. instruments) that can incrementally grow or shrink depending on the context resulting in a discrete sequence of sound events, 4) extraction of statistical regularities using hierarchical N-grams (Pfleger 2002), 5) prediction of continuation, and 6) sonification. The system is tested on voice recordings. We assess the robustness of the performance with respect to complexity and noise of the signal. Given that the number of estimated timbre classes is not necessarily the same as in the ground truth, we propose a performance measure (F-recall) based on pairwise matching. Finally, we sonify the predicted sequence in order to evaluate the system from a qualitative point of view. We evaluate separately the different steps in the process and finally the system as a whole as well as the interacting components of the complete system. Onset detection performs with an F-measure of 98.6% for a data set of a singing voice. Clustering in isolation yields an F-recall of 88.5%. Onset detection jointly with Clustering achieve an F-recall of 91.4%. The prediction of the entire system yields F-recall of 51.3%.

INTRODUCTION

The human music listener distinguishes himself in a number of crucial aspects from a traditional computer system to analyze musical content. The differences manifest themselves in particular through the learning paradigm, the flexibility, and the role of expectation underlying the music analysis process. We enjoy music even if we did not have the privilege to be trained by a merciless *solfège* teacher. Implicitly, we understand the musical structure without being able to name a sound phenomenon as a 5/4 meter, a dominant seventh chord or a *Flügelhorn*. On the other hand, a common music information retrieval (MIR) system needs to be taught by a large quantity of labeled training examples, to learn explicitly what a 5/4 meter, a dominant seventh chord, or a *Flügelhorn* is. We all can follow the brass section in a big band. And if we listen more carefully we will be able to distinguish between a trombone, a trumpet, and a *Flügelhorn* solo. On the other hand, a traditional music information retrieval system that is trained to only recognize a trumpet and a trombone will be challenged when confronted with a *Flügelhorn*. It does not have the flexibility to recognize the *Flügelhorn* as a novel musical instrument sound. Listening to Mozart, a dominant seventh chord invokes the expectation in us to hear a tonic chord and a half-finished *Flügelhorn* melody will make us believe that the melody will be continued by the *Flügelhorn*. On the contrary, traditional context-free music information retrieval systems classify an event as a dominant seventh chord or a *Flügelhorn* without considering previous sounds. In this paper, we will introduce a system prototype that learns in an unsupervised adaptive manner and that generates predictions from an audio sequences. From the first note on it will generate reasonable predictions without using previous knowledge.

We will review relevant previous work.Many approaches to predict musical sequences are based on symbolic representation (Assayag and Dubnov 2004, Lartillot et al. 2001, Mozer 1994, Pachet 2003, Pearce and Wiggins 2004). Paiement et al. (2009) present a model that is capable of predicting and generating melodies using a combination of Bayesian networks, clustering, rhythmic self-similarity and a special representation of melody. The method exploits the self-similarity of a piece and the dyadic organization of its rhythmic structure. Then the occurring distances between rhythmical patterns are clustered. The continuation of a melody is predicted conditioned on the chord root, chord type, and Narmour group of recent melodic notes. Hazan et al. (2009) build a system for generation of musical expectation that operates on music in audio data format. The auditory front-end segments the musical stream and extracts both timbre and time description. In an initial bootstrap phase, an unsupervised clustering process builds up and maintains a set of different sound classes. The resulting sequence of symbols is then processed by a multi-scale technique based on n-grams. Model selection is performed during a bootstrap phase via the Akaike information criterion. Marchini and Purwins (2010) present a non-adaptive system that learns rhythmic patterns from drum audio recordings and synthesizes music variations from the learned sequence. The procedure uses a fuzzy multi-level representation. Moreover, a tempo estimation procedure is used to guarantee that the metrical structure is preserved in the generated sequence.

We will give an overview of the system, introduce its components, namely segmentation, timbre representation, discretization, and prediction. Then we will introduce the F-recall to evaluate the clustering. We will test the performance of the n-gram under noisy conditions. Given a simple singing voice data set, we will test each system module separately and in conjunction. Finally, will give some examples to demonstrate the potential of the system.

SYSTEM OVERVIEW

The system that we present in this paper consists of three main blocks: the preprocessing module, the clustering process and the prediction generator. In the following sections we explain in depth each of these modules.

Segmentation

This section and the following section are dedicated to the problems of audio event temporal localization and representation. This is the entry point to the autonomous listening system we propose and can be considered as the perceptual stage. Event representation is a stage dedicated to isolate and transform the audio events into a feature space. In this space events that are perceptually similar should be numerically similar and events that are perceptually distant should be distant. The preprocessing module is implemented by using an onset detector followed by a feature extractor.

We have used the complex domain based onset detector (Duxbury et al. 2003) since it can be considered as a generalization of onset detection algorithms based on energy, spectral difference, or phase. This onset detection function captures "energetic" percussive onsets as well as soft, "tonal" onsets such as in voice.

In order to determine the onsets, we first process the audio signal frame-by-frame. For each frame, the short-term Fourier transform yields a complex spectrum

$$X_k(l) = R_k(l)e^{j\phi_k(l)}, \tag{1}$$

where R_k and ϕ_k are the magnitude and phase for the k-th bin of frame l with frame length K $(0 \leq k \leq K-1)$. Following Duxbury et al. (2003), we build the onset detection function as the accumulated Euclidean distance between the actual complex spectrum $X_k(l)$ at bin k and the estimated complex spectrum

$$\hat{X}_k(l) = \hat{R}_k(l)e^{j\hat{\phi}_k(l)}, \tag{2}$$

where the estimated amplitude $\hat{R}_k(l)$ is set equal to the magnitude of the previous frame $|X_k(l-1)|$, and the estimated phase $\hat{\phi}_k(l)$ is calculated as the sum of the previous phase and the phase difference between the two preceding frames:

$$\hat{\phi}_k(l) = \text{princarg}\left[\tilde{\varphi}_k(l-1) + (\tilde{\varphi}_k(l-1) - \tilde{\varphi}_k(l-2))\right], \tag{3}$$

where the $\tilde{\varphi}$ operator denotes phase unwrapping and the princarg operator maps the unwrapped value back to the $(-\pi, \pi]$ range.

Then we calculate the bin-wise Euclidean distance between the actual and the estimated complex spectrum, quantifying the stationarity for the k-th bin as:

$$\Gamma_k(l) = \left|X_k(l) - \hat{X}_k(l)\right| \tag{4}$$

$$= \sqrt{\left(\Re(\hat{X}_k(l)) - \Re(X_k(l))\right)^2 + \left(\Im(\hat{X}_k(l)) - \Im(X_k(l))\right)^2}. \tag{5}$$

Summing across all bins and across $M+1$ consecutive frames we yield the onset detection function for frame l:

$$\eta(l) = \sum_{j=\lceil \frac{-M}{2} \rceil}^{\lceil \frac{M}{2} \rceil} \sum_{k=0}^{K-1} \Gamma_k(l+j). \tag{6}$$

The onsets are selected by performing valley picking on the onset detection function. In this study we introduce a custom thresholding algorithm in order to avoid bursts or false positives, which are critical to further processing such as classification or sequence pattern matching. A corner picking algorithm has been designed in order to capture the attack as part of the audio event. Similarly to Bello and Sandler (2003) , the threshold function introduced here is a dynamic adaptive threshold. However in our case, since the main goal is the detection of a corner or a valley, the median of a window in the future is used as threshold. It is comparable to the short-term prediction on the onset detection function:

$$\hat{\eta}(l) = C(l) \cdot \text{median}_{n \in [l, l+H_i]}(\eta(n)) \tag{7}$$

where $C(l)$ is a predefined weighting parameter that controls the sensitivity of the onset detector. The gain $C(l)$ acts inversely that in other onset picking methods, the higher it is the more sensitive to an onset it becomes.

Once we have the short-term prediction of the onset detection function we can perform the difference of both and then the problem becomes finding bumps in a prediction error function:

$$\mu(l) = \eta(l) - \hat{\eta}(l) \tag{8}$$

The prediction error function is then smoothed by a moving average window whose length P determines the temporal masking of onsets.

$$\mu_s(l) = \sum_{n=l-P/2}^{l+P/2} \mu(n) \tag{9}$$

The bump picking algorithm acts by finding positive regions in the prediction error function $\mu_s(l)$ and picking as possible onset candidates one point per region. This method avoids bursts since regions with lots of consecutive changes will lead to long bumps resulting in one single onset candidate. Finally a silence threshold is set and only onset candidates whose onset detection function values are higher than the given threshold are considered as onsets.

Timbre Representation

The feature extractor works by analyzing a short window after the onset. The feature extractor targets to model the timbre of the musical event attacks. The timbre model is performed by analyzing the short audio window through a triangular windowed Mel-scale filterbank followed by a DCT in both the frequency and temporal dimensions. (Mermelstein 1976) The goal of the DCT is to create a sparse representation of the frequency and temporal envelopes and concentrate most of the energy in the first coefficients. This allows us to have a low dimensionality feature vector.

Generation of Discrete Event Sequence by Cobweb

The clustering unit receives multivariate feature vectors from the preprocessing unit and converts them into symbols. Furthermore, the relations between these symbols are expressed in a taxonomy. It is important to state that in our system the events are clustered in an online manner and in order of arrival, since this symbolic representation is used immediately to create predictions on future events.

For this purpose, in Marxer et al. (2007) the Cobweb Fisher (1987) has been used. Cobweb is an incremental clustering model which continuously builds a knowledge tree (hierarchical partitioning of the object space) and assigns to each instance a partition created at each level until the object reaches the leaves of the tree. Each node of the tree represents a concept. A concept is modeled by a univariate Gaussian for each feature dimension. The edges of the structure represent taxonomic relations. Further works McKusick and Thompson (1990), Yoo and Yoo (1995) have proposed techniques to create, in an unsupervised manner, the concept tree based on the sequence of data presented, by the use of a heuristic function to be maximized. The heuristic function used in this paper is the numerical version of the standard category utility function used by Fisher and introduced by Gluck and Corter (1985). Such a version of the Cobweb was presented in McKusick and Thompson (1990) as Cobweb/3 and later extended by Yoo and Yoo (1995) as Cobweb/95. The version presented is Cobweb/3 allows to use real-valued attributes as input and to control the specificity of the partitioning. The heuristic function for a node given its children C_k is defined as:

$$CU_{numeric} = \frac{\sum_k P(C_k) \sum_i \frac{1}{\max(\sigma_{ik}, a)} - \sum_i \frac{1}{\sigma_{iP}}}{4K\sqrt{\pi}}, \qquad (10)$$

where K is the number of children nodes, σ_{ik} is the standard deviation for attribute i in node k, and σ_{iP} is the standard deviation for attribute i in the node, i.e., the no-class membership case for the given level. $P(C_k)$ is the probability for an instance to be classified in children node C_k. This value is calculated as n_k/n where n_k is the instance count in child k and n is the instance count in the node. This formulation adds a new parameter a, the threshold at which to bound the standard deviation of the attributes per class σ_{ik} in order to avoid divisions by zero. This parameter is referred to as the acuity and it controls the resolution of discrimination, i.e., the minimum standard deviation taken into account.

The incorporation of an object is a process of classifying the object by descending the tree along an appropriate path, updating counts along the way, and possibly performing one of several operations at each level. These operators are:

- creating a new node,
- removing all children from a node (pruning),
- combining two classes into a single node, and
- splitting a node into several nodes.

While these operations are applied to a single object set partition (i.e., set of siblings in the tree), compositions of these primitive operations transform a single classification tree. As a search strategy we use hill-climbing through a space of classification trees.

Thereby, input is converted into a sequence of not only symbols, but also of meta symbols (partitions) according to their parent nodes and grand parent nodes in the cobweb tree. The symbols and meta symbols provide the alphabet on which expectations will be generated by the hierarchical N-gram.

We modify the set of possible Cobweb operations (see above) in order to achieve persistent partitioning. This reduced set of operations can perform any of Cobweb's original operations. We reformulate the second Cobweb operation (see above) in order to control the clustering only by new incoming events. Other partitions and past events should not be considered. This reduces the operations to:

- creating a new partition inside a *container partition*, given a *set of contained partitions* which may be empty
- removing a partition, reparenting it's children if it has any

Prediction of Continuation by Hierarchical N-Grams

This section is dedicated to the model of expectation that is based on n-gram encoding of sequence data. We aim at predicting symbol c_t at time t based on previous symbols $c_{t-n+1}, \ldots, c_{t-1}$. The output of the symbol grounding process in the previous section yields a stream of symbols linked to nodes on a taxonomy tree. Hierarchical n-grams (Pfleger 2002) are combinations of sparse n-gram models in a hierarchical structure that allows compositional learning. Compositional learning consists in learning long patterns from already learned sub-patterns. In sparse n-grams counts of the most frequent patterns and a separate total count for the non-frequent patterns are kept. This technique separates the estimates of patterns whose statics are reliable from the estimates of infrequent patterns whose statistics are biased. On the other hand, the multi-width exhaustive approach consists in keeping the count of all possible patterns of at most length N. These models are able to represent any distribution of patterns up to width N. The correctness of the statistics estimates is proportional to the count of repetitions of the pattern. Larger patterns take longer to reduce their variance error, since due to their length they occur more rarely.

Be $\mathscr{C}_1 = \{c^1, \ldots, c^{|\mathscr{C}_1|}\}$ the set of cluster indices, renumbered so that they reflect the order of their first appearance in the symbol sequence $\mathbf{c} = (c_1, \ldots, c_t)$, achieved from the discretization process in the previous section. $\mathscr{C}_1$ forms the alphabet of the n-gram. Then, $\mathscr{C}_N$ is the set of all possible N-grams of length N composed from alphabet $\mathscr{C}_1$. For exploiting sparsity, we only consider the patterns that have actually occurred as a subsequence of $\mathbf{c}$ so far until time t. This set we denote by $\mathscr{C}_n = \{\mathbf{c}^1, \ldots \mathbf{c}^{|\mathscr{C}_n|}\}$, in which again the length n sub-patterns are ordered according to their first appearance. $o(\mathbf{c})$ is defined as the position of $\mathbf{c}$ in $\mathscr{C}_n$. We consider hierarchical n-grams of maximal length N. Be $C_{n,i} (n \leq N)$ the frequency count of the i-th pattern of length n and be $T_{n,i}$ the total count of patterns of length n since pattern i has occurred for the first time. In Algorithm 1, we use the counts $T_{n,j}$ and $C_{n,j}$ in order to iteratively calculate the joint probabilities $P(\mathbf{c}^i)$ of all patterns $\mathbf{c}^i$ of all length $0 \leq n \leq N$ occurred so far.

The hierarchical n-gram method borrows ideas from the sparse and multi-width exhaustive methods. In the hierarchical n-gram method the probability distribution of a pattern of length n (Equation 14) uses statistics estimates of the sub-patterns of lengths $n-1$ (Equation 13) or smaller weighted by their confidence. The confidence values are based on the number of occurrences of these sub-patterns. Therefore when a pattern of length n has appeared rarely in the data stream, its probability of occurrence is estimated from a small number of counts and it is not reliable. In this case the probability of appearance is better estimated from the $n-1$ length sub-patterns. In other words, the information of patterns of large lengths is integrated with the information of models of small lengths to reduce the bias error and the variance error respectively. Pfleger shows that the probability of a given pattern can be calculated in a linear sweep by updating all the probabilities in order of pattern's first

occurrence and length.

In order to adapt Pfleger (2002)'s hierarchical *n*-gram to our architecture, we have to link the operations of the clustering model to the operations on the *n*-gram (Figure 1). When two or more clusters are merged in the clustering model, we have to remove the superfluous clusters from the set of cluster indices (Equation 11) and to sum up the counts for the merged clusters (Equation 12). For example, if the *n*-gram tracks patterns *bbc* and *bbd* and suddenly the clustering model merges symbols *c* and *d* into a new symbol *e*, the *n*-gram must sum up the counts of *bbc* and *bbd* and substitute them by the counts of *bbe*.

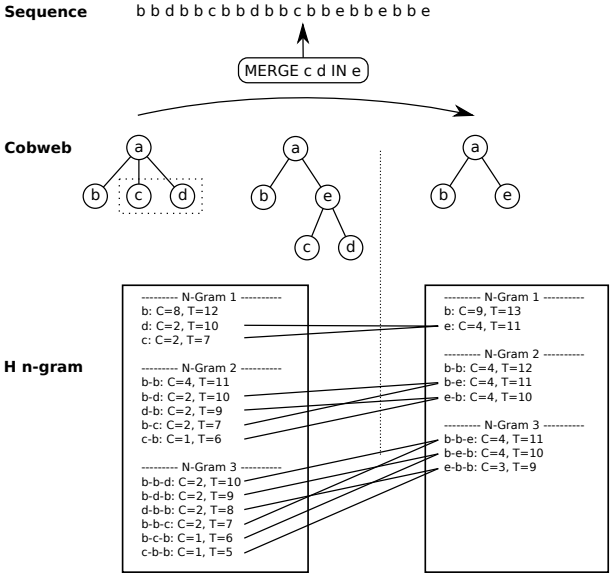

Figure 1: The Effect of a concept merge in the hierarchical n-gram. Nodes *c* and *d* are merged into the new symbol *e*. The *n*-gram inherits the counts for patterns including *c* and *d* to patterns including *e*.

PERFORMANCE ANALYSIS OF THE SYSTEM

Evaluation Measures

Measures for Clustering Evaluation

For a feature space $\mathscr{X}$, let $\mathscr{I} = \{1,\dots n\}$ be a set of annotated labels and $a : \mathscr{X} \to \mathscr{I}, x \to a(x) = i$ an annotation function, $\mathscr{J} = \{1,\dots m\}$ a set of cluster indices, $c : \mathscr{X} \to \mathscr{I}, x \to c(x) = i$ a clustering assignment function. In order to discuss measures to evaluate clustering we define the set $\mathscr{A}'_i$ of feature vectors x that are annotated with label i as $\mathscr{A}'_i = \{x \in \mathscr{X} : a(x) = i\}$. Analogously, we define the set $\mathscr{C}'_j$ of feature vectors x that are assigned to a cluster with cluster index j by $\mathscr{C}'_j = \{x \in \mathscr{X} : c(x) = j\}$. Besides entropy and purity (Zhao et al. 2005), F-Measure has been suggested by Larsen and Aone (1999) to evaluation clustering algorithms:

$$F = \frac{1}{|\mathscr{X}|} \sum_{i=1}^{n} |\mathscr{A}'_i| \cdot \max_{1 \le j \le m} \frac{2\mathscr{A}'_i \mathscr{C}'_j}{\mathscr{A}'_i + \mathscr{C}'_j}. \tag{15}$$

Pairwise F-Measure

The classic F-Measure which is used in many onset detection (Downie et al. 2005, Leveau and Daudet 2004) and supervised clustering (Paulus and Virtanen 2005) evaluations, is not suited for the transcription and prediction tasks under the assumption of an undetermined number of clusters. This is due to the fact that for these tasks we might end up with a different number of clusters than classes and no specific mapping about which cluster represents which class. Therefore, we have reformulated the problem in order to analyze the clustering decisions in a pairwise manner. This means that instead of evaluating the correct clustering of the individual symbols, we evaluate the clustering decisions of the symbols two by two. Under the above reasoning we propose a brute force method which checks all possible pair combinations of the symbols in the input sequence. The check consists in counting the number of pairs that were estimated equal and were annotated equal as well as counting the number of pairs that were annotated different and estimated different. For feature space $\mathscr{X}$, we apply the common notions of recall to pairwise data $\mathscr{P} = \{(x,x') \in \mathscr{X} \times \mathscr{X} : x \neq x'\}$. Let $\mathscr{A} = \{(x,x') \in \mathscr{P} : a(x) = a(x')\}$ be the set of feature vector pairs (x,x') that are annotated with the same label. Let $\mathscr{C} = \{(x,x') \in \mathscr{P} : c(x) = c(x')\}$ be the set of feature vector pairs (x,x') that are assigned to the same cluster. Then define the pairwise recall $R_{\mathscr{A},\mathscr{C}}$ as

$$R_{\mathscr{A},\mathscr{C}} = \frac{|\mathscr{A} \cap \mathscr{C}|}{|\mathscr{A}|}. \tag{16}$$

We apply recall also to the complementary sets $\bar{\mathscr{A}} = \mathscr{P} \backslash \mathscr{A}$ and $\bar{\mathscr{C}} = \mathscr{P} \backslash \mathscr{C}$. Then we define the pairwise F-Recall as

$$F = \frac{2 \cdot R_{\mathscr{A},\mathscr{C}} \cdot R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}}{R_{\mathscr{A},\mathscr{C}} + R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}}. \tag{17}$$

These measures can be better understood by looking at limit situations. When all the input symbols get clustered in one single cluster, the probability of having pairs of symbols of different classes in separate clusters is null, therefore $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}} = 0$ would be 0. However the probability of finding pairs of symbols of same classes in the same cluster is maximal, this means that $R_{\mathscr{A},\mathscr{C}} = 1$. On the other hand when each of the input symbols gets estimated in a separate cluster the probability of finding a pair of symbols of equal classes in the same cluster is null, therefore $R_{\mathscr{A},\mathscr{C}} = 0$.. But the probability of finding pairs of symbols of different classes in separate clusters is maximal, which would imply a precision of $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}} = 1$.

Data Sets

Two main sets of tests have been carried out in this study. The first set of tests consists on evaluating the theoretical and practical limits of the algorithms, by creating synthetic data sets with high control over the generative parameters and evaluations.

The second set of tests is based on real life data by using the following data set:

Voice Informal low quality recordings that serve as proof of concept, to the unsupervised nature and adaptability of the system.

Tests with Synthetic Data

In this section we present a series of tests based on synthetic (generated) data to evaluate the performance of the different clustering and expectation techniques presented in this study. These tests help determine the actual limits of the algorithms in use.

Expectation

This test consists in feeding the sequence learning algorithm with a sequence of symbols and querying for the expected next symbol. The sequence of symbols is generated by selecting a pattern and repeating it multiple times. Different types of noise are added to the sequence in order to tests the limits of the sequence learning techniques:

Algorithm 1 The Hierarchical N-Gram for Merged Clusters

Initialization $\mathscr{C} = \{\}$
for incoming event c_t **do**
  for $1 \leq n \leq N$ **do**
    if $(c_{t-n+1}, \ldots, c_t) \notin \mathscr{C}_n$ **then**
      Add new pattern: $\mathscr{C}_n = \mathscr{C}_n \cup (c_{t-n+1}, \ldots, c_t)$, $T_{n,|\mathscr{C}_n|} = 1, C_{n,|\mathscr{C}_n|} = 1$
    else if $\mathbf{c}^1, \ldots, \mathbf{c}^k \in \mathscr{C}_n$ are merged by Cobweb **then**

$$o' = \min(o(\mathbf{c}^1), \ldots, o(\mathbf{c}^{\mathbf{k}})), \mathscr{C}_n = \mathscr{C}_n \backslash \{\mathbf{c}^1, \ldots, \mathbf{c}^{o'-1}, \mathbf{c}^{o'+1}, \ldots, \mathbf{c}^k\} \tag{11}$$

$$C_{n,o'} = \sum_{i=1}^{k} C_{n,o(\mathbf{c}^k)} \tag{12}$$

    else
      Update Counts: $T_{n,o(c_{t-n+1},\ldots,c_t)} = T_{n,o(c_{t-n+1},\ldots,c_t)} + 1$
      $C_{n,o(c_{t-n+1},\ldots,c_t)} = C_{n,o(c_{t-n+1},\ldots,c_t)} + 1$
    end if
  end for
  Calculate joint probabilities:
  $P(\{\}) = \frac{1}{|\mathscr{C}_n|}$
  for $1 \leq n \leq N$ **do**
    for $\mathbf{c} \in \mathscr{C}_n$ **do**

$$Q(\mathbf{c}) = Q(c_1, \ldots, c_n) = (1 - \sum_{k=1}^{o(\mathbf{c})} P(\mathbf{c}^k)) \tag{13}$$

$$P(\mathbf{c}) = \frac{1}{T_{1,1}} \left[C_{n,o(\mathbf{c})} + \sum_{j=0}^{o(\mathbf{c})-1} (T_{n,j} - T_{n,j+1}) \cdot Q(\mathbf{c}) \cdot \frac{P(c_2, \ldots, c_n)}{Q(c_2, \ldots, c_n)} \right] \tag{14}$$

    end for
  end for
end for

Switching noise In the original sequence with a given probability, a symbol is exchanged for a random other one with a given probability.

Skipping noise In the original sequence with a given probability, a symbol gets skipped with a given probability.

Running computer simulations we investigated learning rate and convergence of the algorithm with respect to the following parameters:

Pattern length The length of the basic pattern is varied, maintaining the number of repetitions fixed.

Pattern repetitions The number of repetitions is varied, whereas the length of the basic pattern remains constant.

For the evaluation we applied F-Recall measure to the entire sequence.

The inherent ambiguity of a sequence is crucial for the difficulty of learning this particular sequence. E.g. in the sequence *'abcdabcd'* each symbol follows the other deterministically: after an *'a'* there is always a *'b'* and after a *'b'* there is always a *'c'*. On the contrary, in sequence *'abacabac'* after an *'a'* a *'b'* can follow as well as a *'c'*. The continuation of a sequence is therefore more ambiguous, considering only the previous symbol. To evaluate the algorithm on all possible length n sequences $\mathbf{c} \in \mathscr{C}_n$ composed of the alphabet $\mathscr{C}$ would be computational excessive. Therefore, for $\mathbf{c} = (c_1, \ldots, c_n)$ we consider all possible partitions $\mathscr{P}$ of the index set $\{1, \ldots, n\}$, where a partition is defined as a set $\mathscr{P}^i = \{\mathscr{P}^i_1, \ldots, \mathscr{P}^i_j\}$ of subsets $\mathscr{P}^i_j \subset \{1, \ldots, n\}$ so that $\cup_j \mathscr{P}^i_j = \{1, \ldots, n\}$ and $\mathscr{P}^i_j, \mathscr{P}^i_{j'}$ disjoint for $j \neq j'$. To give an example: The index set $\{1,2\}$ would have the following two partitions: $\{\{1\},\{2\}\}$ and $\{\{1,2\}\}$. For a given partition $\mathscr{P}^i$ we define $X_{\mathscr{P}^i}(l) = \mathscr{P}^i_j \iff l \in \mathscr{P}^i_j$. Thus, $X_{\mathscr{P}^i}(l)$ is the part of the partition to which index l belongs to. We define the set of patterns belonging to partition $\mathscr{P}^i$ as $\{\mathbf{c} : c_k = c_l \iff X_{\mathscr{P}^i}(k) = X_{\mathscr{P}^i}(l), \forall k \neq l\}$. Thus, the set of patterns with equal symbols at indices in the same part and different symbols at indices that belong to different parts. For example the partition $\{\{1,3\},\{2\},\{4\}\}$ of a sequence of length $n = 4$ would contain patterns such as *'abac'* or *'bcba'*, where only indices 1 and 3 have the same symbol.

Every pattern belonging to the same partition will give the same result. Therefore all sequences of the same partition can be considered equivalent with respect to our n-gram. So for a length n sequence, we only need to test the system for one pattern per partition.

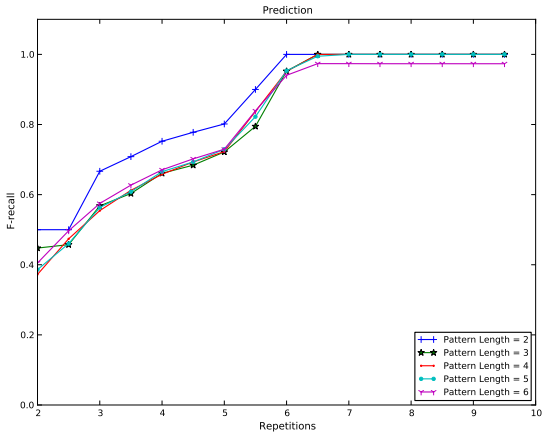

Figure 2: F-recall (Equation 17) of the sequence learning process on patterns of different lengths given the number of pattern repetitions.

Figure 2 shows the results of the test of running the expectation system on multiple repetitions of patterns of different lengths. We measure the F-recall of the last 5 repetitions of the pattern at different points in the sequence. For this test the

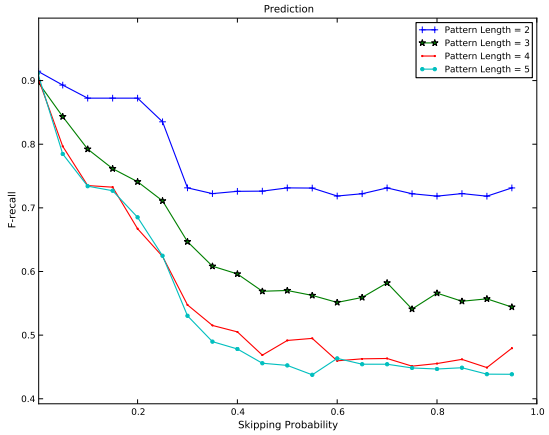

Figure 3: F-recall (Equation 17) values of the sequence learning process on 20 repetitions of patterns of different lengths given the skipping probability.

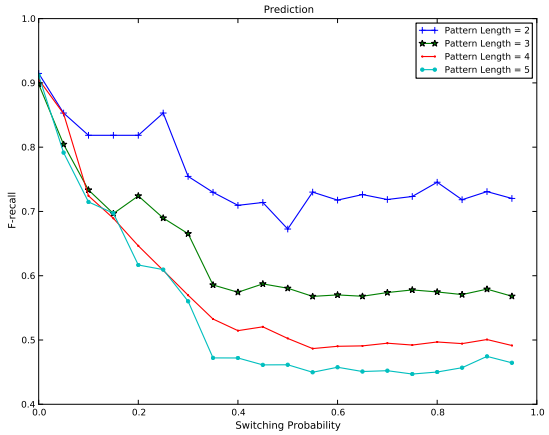

Figure 4: F-recall (Equation 17) of the sequence learning process on 20 repetitions of patterns of different lengths given the switching probability.

n-gram model is set to a maximum n-gram length of 5. We see how the sequence learning process is capable of reaching perfect prediction for lengths up to 5, after 7 repetitions of the pattern. Since we are evaluating a window that contains the last 5 repetitions of the pattern, we are actually reaching perfect prediction after 2 repetitions of the pattern, for lengths under the maximum pattern length tracked by n-gram. For patterns larger than the ones tracked by the n-gram perfect prediction cannot be achieved, due to the inherent ambiguities of some of the possible patterns that must be coded somehow in the n-gram model. In these cases the model should track longer patterns to disambiguate between the possible transitions. Several solutions for this problem are proposed in Pfleger (2002).

In Figure 3 we can see the effects of skipping errors in the input signal. In our case skipping occurs when the onset detection misses an event. We measure how the F-recall measure is influenced by the skipping probability. In this test the evaluation is performed on the whole sequence for 20 repetitions of the pattern. Apart from patterns of length 2, all other lengths react similarly to skipping. They mostly degrade linearly until a skipping probability of 0.5 is reached. As we see in the next section the onset detection process is quite reliable in terms of skipping on the specific data set that we used.

Finally Figure 4 we evaluate the effect of clustering errors on the sequence learning process. We test the sequence learning F-recall performance measure against different probabilities of switching a symbol by a different one. Similarly to the case of skipping, the F-recall value degrades up to a switching probability of 0.4.

Tests with Audio Recordings

We will test each process stage separately.

Onset detection

For the evaluation of the onset detection we employ a widely used procedure Downie et al. (2005), Leveau and Daudet (2004). As a reference serve the onset times, manually annotated by (a) subject(s). The onsets estimated by the onset detection algorithm are then compared to the manually annotated onsets. An annotated and estimated onsets are considered a match when their difference in time is smaller than a given threshold. In our evaluation we use an onset match threshold of 50 ms. Since the data is assumed monophonic, the evaluation only permits a one-to-one mapping between estimated and annotated onsets.

Table 1: Onset detection: F-measure, (precision and recall) for different thresholds $C(l)$ from 7 (rows) and smoothing length M from 6 (columns) on the *Voice* data set.

	21	25	29
0.85	97.8 (95.9, 100.0)	98.6 (97.4, 100.0)	98.6 (97.4, 100.0)
0.9	97.0 (94.6, 100.0)	**98.6 (97.4, 100.0)**	98.6 (97.4, 100.0)
0.95	95.8 (92.4, 100.0)	97.8 (95.9, 100.0)	98.6 (97.4, 100.0)

As we can see from the results in Table 1, the *Voice* data set does not pose a challenge to the onset detection method. Therefore, we focus on the clustering and the prediction stage. We also notice that for smoothing lengths larger than 25 the system does not improve significantly. Large smoothing lengths reduce the temporal precision of onsets, which is important to good feature extraction, since most information of the event is located in the attack.

Clustering

We now evaluate the performance of the incremental classification of musical events. In order to assess the clustering process in isolation we assume error-free onset detection on the previous stage. In order to achieve this, we use the annotated onsets as input.

Table 2: Clustering: F-recall (Equation 17), ($R_{\mathscr{A},\mathscr{C}}$, $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}$) for different timbral acuities a from 10 (rows) and analysis window lengths (columns). *Voice* data set.

	75	125	175
16	48.9 (56.0, 84.9)	82.4 (94.1, 75.8)	77.4 (96.0, 69.3)
17	51.5 (55.8, 88.9)	84.2 (93.0, 78.9)	79.8 (96.0, 72.2)
18	36.1 (39.8, 93.6)	85.8 (93.0, 81.5)	83.3 (96.0, 76.5)
19	37.4 (39.8, 95.5)	**88.4 (91.4, 86.3)**	82.4 (95.0, 75.6)
20	37.7 (38.3, 97.1)	70.7 (74.8, 87.3)	83.0 (95.0, 76.4)

The Table 2 shows high F-recall measure which mean that the timbre model and clustering process can successfully classify the audio events. We also notice that short analysis window lengths (75ms) do not capture enough of the characteristic part of the sound to correctly classify them. The low precision and high recall values implies that the system is wrongly clustering most events together. This could be due to the fact that the first 75ms after the onsets are similar for all events, and most information of the event is located between 75ms and 125ms

after the onset. This test, as explained above, was performed using the annotated onsets. The results could change when the onsets are estimated. This effect is evaluated in the transcription test.

Expectation

The expectation test addresses the evaluation of the performance of the sequence learning module on the data sets. The test consists in evaluating how well the sequence learning algorithm is able to predict the next musical event c_t after the realized events $c_1, \ldots, c_{t-1}$. We assume that sequence $c_1, \ldots, c_{t-1}$ has been correctly identified by using the manually annotated symbols. We compare the symbol $\hat{c} = \text{argmax}_i P(c_1, \ldots, c_{n-1}, i)$ predicted by the sequence learner to the actual next event c_t.

Table 3: Expectation: F-recall (Equation 17), ($R_{\mathscr{A},\mathscr{C}}$, $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}$) for different maximum lengths of the *n*-gram (rows) for the *Voice* data set.

	hngram
3	50.8 (50.8, 65.2)
4	64.5 (63.8, 66.3)
5	**69.4 (68.3, 71.6)**
6	69.4 (68.3, 71.6)
7	69.4 (68.3, 71.6)

Results in Table 4 show that the sequences of the *Voice* data set are actual predicted quite successfully using our hierarchical *n*-gram model. We can also see that for *n*-gram maximum lengths higher than 5 the result does not improve.

In a similar manner we could also evaluate the prediction of the next note onset based on the previous inter-onset intervals. However, here we limit our focus on the prediction evaluation of the next event disregarding its exact onset time. Below, assess the overall performance of the entire system including timing information.

Transcription

The transcription test evaluates the subsystem composed of onset detection, feature extraction, and clustering. The test was conducted on the *Voice* data set. In Table 4, the detected events were compared to the annotated labels, the ground truth, using pairwise F-recall (Equation 17). In order to assess the stability of the system we have tested it with different values of the two most sensible parameters involved in the task: the window length and the acuity. The analysis window length refers to the frame length immediately following the detected onset. From this analysis frame the feature vector is extracted. The acuity a refers to the parameter controlling the resolution of the Cobweb algorithm, see 10.

Table 4: Transcription: F-recall (Equation 17), ($R_{\mathscr{A},\mathscr{C}}$, $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}$) of acuity a from 10 for timbre clustering (rows) versus analysis window length (columns) measured on the *Voice* data set.

	75	125	175
18.5	35.3 (39.2, 92.5)	91.4 (93.5, 90.5)	83.8 (100.0, 74.5)
19	36.0 (39.2, 93.5)	90.3 (91.9, 89.7)	86.2 (100.0, 77.5)
19.5	36.0 (39.2, 93.5)	**91.4 (89.5, 94.9)**	83.4 (93.0, 77.5)
20	36.0 (39.2, 93.5)	91.0 (88.6, 94.9)	82.1 (85.0, 82.1)
20.5	38.7 (39.2, 98.3)	84.1 (81.6, 96.7)	84.2 (85.0, 85.7)

The transcription results in Table 4 resemble the results from the clustering test. This is expected since the onset detection process performs almost perfectly on the data set. However, the onset detection evaluation considers a detected onset correct if it is within a temporal interval from an annotated onset which we set to be 50 ms. Given that the results of our transcription test are almost the same, we can conclude that the onset position difference between the annotations and the estimations will not influence significantly the feature extraction and clustering process.

Prediction

The prediction task consists in running the full system. After the transcription of the events $c_1, \ldots, c_{t-1}$ the system calculated a prediction $\hat{c}_t$ of the next event c_t. This prediction includes the next symbol (derived from the clustering) and its timing. The threshold for matching predicted events from annotated events is larger in this task. We have set this tolerance threshold to 150 ms.

Table 5: Full Prediction: F-recall (Equation 17), ($R_{\mathscr{A},\mathscr{C}}$, $R_{\bar{\mathscr{A}},\bar{\mathscr{C}}}$) for different temporal acuities a from 10 (rows) and timbral acuities a from 10 (columns). *Voice* data set.

	19	20	21
0.0125	35.0 (42.9, 30.2)	36.9 (38.4, 36.0)	24.0 (24.5, 42.3)
0.025	41.5 (48.6, 36.8)	43.8 (45.0, 43.5)	27.4 (28.2, 47.5)
0.0375	41.7 (49.7, 36.6)	44.5 (46.7, 42.9)	31.1 (32.0, 48.1)
0.05	42.9 (50.7, 38.3)	44.9 (46.6, 43.8)	25.9 (26.6, 49.2)
0.0625	48.2 (56.7, 43.1)	**51.3 (53.6, 49.4)**	33.1 (33.3, 54.2)
0.075	45.7 (53.7, 41.0)	48.2 (50.0, 46.7)	30.4 (29.9, 52.1)

Examples

In this section, we present a few examples of transcription and prediction in order to assess the performance, evolution and shortcomings of the system. We have calculated the matching matrix between the annotated onset events ('score') of a class and the detected onsets of a cluster (Hazan et al. 2009). In this matching matrix we can iteratively yield the maximal entry, thereby establishing a connection between a row (class) and a column (cluster). After elimination the row and column of the maximal entry we determine the maximal entry again until the matrix vanishes. This procedure endows us with an optimal mapping between the classes and the clusters. In Figures 5 and 6, we display sequences of classes and clusters on the same line if they are interconnected through this mapping.

In Figure 5, we can see the system working at its best. The first three predictions are wrong. The second and third prediction are off in time. Therefore, they do not match to their annotated counterparts. The first three cluster mismatches are expected, since the system has no previous knowledge of the symbol space and of the sequence and therefore cannot predict symbols nor patterns that have not yet occurred. The time deviation errors are due to the fact that the recorded voice does not follow a temporally regular pattern and that it is not able to predict fluctuations in timing. In Figure 6, we observe how the system adapts to pattern changes within the sequence. The beginning of the sequence is wrongly predicted due to errors in the clustering of the sounds. However, after having processed enough sounds the system correctly predicts the next timbre. The errors in the middle of the sequence are due to a pattern change. The *n*-gram is able to update the statistics and perform correct predictions after two occurrences of the new pattern.

CONCLUSION

We have presented a full system that predicts the next sound event from the previous events, operating on audio data. Taking into account no previous knowledge, neither on the used sounds or instruments nor on the timing and rhythmical structure of the audio segment, the system starts from *tabula rasa*, performing predictions from the very first sound event. The system adapts

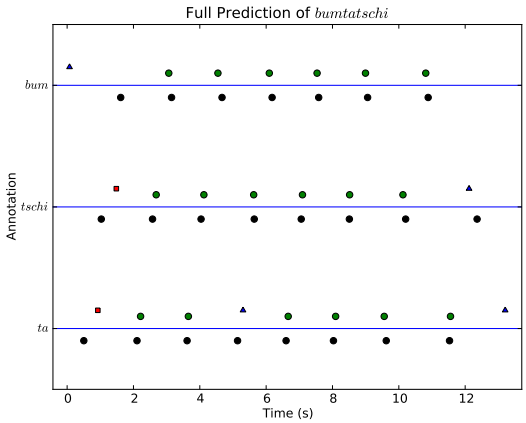

Figure 5: The system quickly captures a simple *ta-tschi-bum* pattern. Time (horizontal axis) is mapped versus event labels (vertical axis). Annotated labels are indicated in black below the lines. Above the horizontal lines we find correctly (green ○), incorrectly (red □), and unmatched (blue △) estimated events.

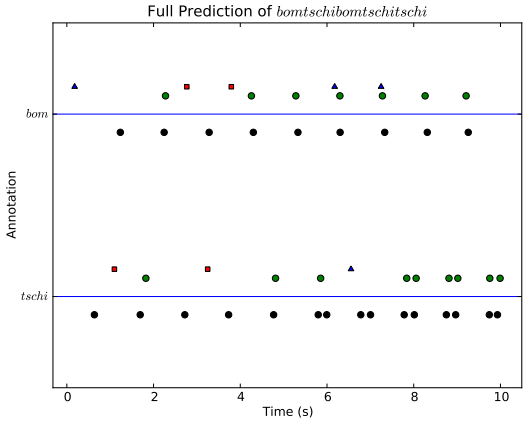

Figure 6: The system adapts to a pattern change from *tschi-bom* to *tschi-tschi-bom*. For the graphical explanation cf. to Figure 5.

to pattern changes in the sequence as well as the appearance of new sounds or instruments at any time. In addition, we have presented the pairwise F-recall, a new measure to evaluate unsupervised clustering. Currently the system is limited by the lack of metrical analysis. This makes it specially sensitive to missed onsets. Being central in the music listening process, the metrical process could significantly improve the quality of predictions.

Inspired by these ideas, we imagine a musical improvisational dialogue between a human and a machine in which the human may spontaneously articulate novel ideas such as new sounds, motifs, rhythms, or harmonies. A dumb and ignorant machine would dampen and finally stop the musical flow. But if the machine could take up the novel idea, reply to it, varying the suggestions of his human partner, both could continue an inspired musical conversation forever . . .

REFERENCES

Gérard Assayag and Shlomo Dubnov. Using Factor Oracles for machine Improvisation. *Soft Computing*, 8(9):604–610, 2004.

Juan Pablo Bello and Mark B. Sandler. Phase-based note onset detection for music signals. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing,ICASSP*, 5(2):441–444, 2003.

J. Stephen Downie, Kris West, Andreas F. Ehmann, and Emmanuel Vincent. The 2005 music information retrieval evaluation exchange (mirex 2005): Preliminary overview. In *ISMIR*, pages 320–323, 2005.

C. Duxbury, J. Bello, M. Davies, and M. Sandler. Complex domain onset detection for musical signals. *Proceedings Digital Audio Effects Workshop (DAFx)*, 2003.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Mach. Learn.*, 2(2):139–172, 1987.

M. Gluck and J. Corter. Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, pages 283–287, 1985.

A. Hazan, R. Marxer, P. Brossier, H. Purwins, P. Herrera, and X. Serra. What/when causal expectation modelling applied to audio signals. *Connection Science*, 21:119 – 143, 06/2009 2009.

B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM, 1999.

Olivier Lartillot, Shlomo Dubnov, Gérard Assayag, and Gill Bejerano. Automatic modeling of musical style. In *Proceedings of the International Computer Music Conference (ICMC 2001)*, La Havana, Cuba, 2001.

Pierre Leveau and Laurent Daudet. Methodology and tools for the evaluation of automatic onset detection algorithms in music. In *In Proc. Int. Symp. on Music Information Retrieval*, pages 72–75, 2004.

Marco Marchini and Hendrik Purwins. Unsupervised generation of percussion sound sequences from a sound example. In *Sound and Music Computing Conference*, 2010.

Ricard Marxer, Piotr Holonowicz, Hendrik Purwins, and Amaury Hazan. Dynamical hierarchical self-organization of harmonic, motivic, and pitch categories. In *Music, Brain and Cognition. Part 2: Models of Sound and Cognition, held at NIPS)*. Vancouver, Canada, 2007.

K. McKusick and K. Thompson. Cobweb 3: A portable implementation. *Technical Report No. FIA-90-6-18-2*, 1990.

P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. In *Pattern Recognition and Artificial Intelligence*, pages 374–388. Academic, New York, 1976.

M.C. Mozer. Neural network music composition by prediction: Exploring the benefits of psychophysical constraints and multiscale processing. *Connection Science*, 6:247–280, 1994.

Francois Pachet. The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–341, 2003.

J.F. Paiement, Y. Grandvalet, and S. Bengio. Predictive models for music. *Connection Science*, 21(2):253–272, 2009.

Jouni Paulus and Tuomas Virtanen. Drum transcription with non-negative spectrogram factorisation. In *Proc. of the 13th European Signal Processing Conference*, Antalya, Turkey, September 2005.

Marcus T. Pearce and Geraint A. Wiggins. Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385, 2004.

Karl Pfleger. *On-line Learning of Predictive Compositional Hierarchies*. PhD thesis, Stanford University, 2002.

Jungsoon Yoo and Sung Yoo. Concept formation in numeric domains. In *CSC '95: Proceedings of the 1995 ACM 23rd annual conference on Computer science*, pages 36–41, New York, NY, USA, 1995. ACM Press.

Y. Zhao, G. Karypis, and U. Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, 2005.