Unsupervised Incremental Learning and Prediction of Music Signals

Ricard Marxer, Hendrik Purwins

***Abstract*—A system is presented that segments, clusters and predicts musical audio in an unsupervised manner, adjusting the number of (timbre) clusters instantaneously to the audio input. A sequence learning algorithm adapts its structure to a dynamically changing clustering tree. The flow of the system is as follows: 1) segmentation by onset detection, 2) timbre representation of each segment by Mel frequency cepstrum coefficients, 3) discretization by incremental clustering, yielding a tree of different sound classes (e.g. instruments) that can grow or shrink on the fly driven by the instantaneous sound events, resulting in a discrete symbol sequence, 4) extraction of statistical regularities of the symbol sequence, using hierarchical N-grams and the newly introduced conceptual Boltzmann machine, and 5) prediction of the next sound event in the sequence. The system's robustness is assessed with respect to complexity and noisiness of the signal. Clustering in isolation yields an adjusted Rand index (ARI) of 82.7% / 85.7% for data sets of singing voice and drums. Onset detection jointly with clustering achieve an ARI of 81.3% / 76.3% and the prediction of the entire system yields an ARI of 27.2% / 39.2%.**

***Index Terms*—Music information retrieval, unsupervised learning, adaptive algorithms, prediction algorithms**

I. Introduction

Human music listening adapts to novel acoustic stimuli and is largely based on unsupervised learning, in contrast to most traditional music analysis systems. For music transcription [9], prediction [8, 36], representation [22, 32], automatic accompaniment, or human-machine-improvisation [2, 34], a traditional system usually is based either on symbolic data instead of audio input, or on classifiers that are pre-trained on a labelled data base [9]. If a system, based on pre-trained classifiers needs to cope with new musical concepts (instruments, harmonies, pitches, motifs) it has not been designed for, it may cease to work reasonably. Such a system would have to be retrained with labeled data, every time a new instrument (pitch, harmony etc.) appears. This presents a severe lack of flexibility of such a system, in contrast to human cognition processing new instruments and harmonies with ease, even if one has not heard them before. A human mind can grasp a novel motif, when listening to a piece or an improvisation. Unsupervised learning (clustering) instead of supervised classification is one paradigm how an algorithm can model the cognition of novel concepts [17, 24, 26]. Based on a discrete representation of the input derived by clustering, an n-gram, i.e. a suffix tree, can be used as a statistical representation of the structure of the input sequence [8, 36]. In this paper, we extend such a system by equipping it with the capability to deal with a varying number of clusters. The number of clusters can increase if a new instrument appears. The cluster number decreases if two instruments become to sound very similar. We implement these features by using unsupervised *online learning*. This requires that the n-gram (suffix tree) must be coupled with the clustering in order to be able to merge or split the symbol counts when cluster numbers change. We introduce a system prototype that learns in an unsupervised, adaptive manner and that generates predictions from audio sequences. From the first note it will begin to generate reasonable predictions without using previous knowledge.

Many previous approaches to predicting musical sequences are based on symbolic representation [2, 8, 22, 32, 34, 36]. Paiement et al. [35] 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 distances between rhythmical patterns are clustered and the continuation of a melody is predicted conditioned on the chord root, chord type, and Narmour group of recent melodic notes. Hazan et al. [17] 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 timing 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 [24] 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. Online clustering has been proposed by Zhang et al. [46] for document clustering. Bertin-Mahieux et al. [5] have used online k-means to cluster beat-chroma patterns. The Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) [43] has been used for segmentation in conjunction with clustering. Fox et al. [13] and Ren et al. [40] have proposed 'sticky' versions of the HDP-HMM that introduce explicit modelling of state occupancy duration. These

R. Marxer is with Speech and Hearing Group, Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello Street, Sheffield, S1 4DP, UK, r.marxer@sheffield.ac.uk.

H.Purwins is with Audio Analysis Lab and Sound and Music Computing Group, Aalborg Universitet København, A.C. Meyers Vænge 15, 2450 Copenhagen SV, hpu@create.aau.dk.

R. Marxer and H. Purwins were with Music Technology Group, Universitat Pompeu Fabra, Roc Boronat, 138, 08018 Barcelona, Spain.

H. Purwins was also with Neurotechnology Group, Berlin Institute of Technology, Sekr. MAR 4-3, Marchstr. 23, 10587 Berlin, Germany.

This work was partially funded by the EmCAP (European Commission FP6-IST, contract 013123) and INSPIRE-ITN (European Commission FP7-PEOPLE-2011-290000) projects. Thanks to Daniel Bartz from Berlin Institute of Technology for advice in statistics and for proof reading.

models are applied to segmentation of a Beethoven sonata into musical sections [40] and to speaker diarization [13]. Stepleton et al. [42] used the block diagonal infinite hidden Markov model for musical theme labelling. However, these methods do not perform incremental online learning, whereas we propose an online incremental clustering method that uses a separate segmentation method (onset detection) and switches relatively rapidly between states. Bargi et al. [3] have adapted HDP-HMM to an online setting employing an initial supervised learning phase (bootstrap) whereas our approach is entirely unsupervised.

A part of the work covered in this paper, the application of the hierarchical n-grams on the *Voice* data, has been presented previously [26]. Here we compare that method with the conceptual Boltzmann machine and with HDP-HMM on an extended data set using a more advanced evaluation measure (the adjusted Rand index) and providing more examples of adaptive clustering. We will give an overview of the system, introduce its components, namely segmentation, timbre representation, clustering, and prediction. Then we will introduce the adjusted Rand index, test the performance of the sequence analysis algorithms under noisy conditions, of each system module separately, and in conjunction. Finally, we will give some demonstration examples. Audio-visual data and examples are available on the supporting website [27].

II. System Overview

The system that we present in this paper (cf. Fig. 1) consists of four main stages: segmentation by onset detection, feature extraction resulting in timbre representation, incremental clustering giving a symbol sequence, and sequence analysis yielding a prediction of the next symbol. In particular, the clustering tree generated by incremental clustering grows and shrinks online, driven by the most recent sounds. In turn, the sequence model adapts to the changing numbers of symbols. Segmentation and representation can be interpreted as a model of perception, whereas discretization and prediction can be considered to be a cognitive model.

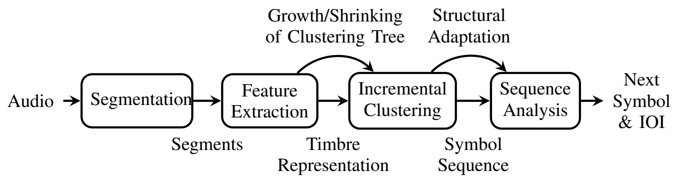

Fig. 1. System architecture: An audio sound file is segmented, using onset detection. Each segment is then represented as a high-dimensional timbre feature vector which is clustered into symbols. Symbols are added or removed to the clustering tree on the fly. The symbol sequence is then statistically analysed, adapting to the varying number of symbols, allowing for prediction of the next symbol and the next inter-onset interval (IOI).

A. Segmentation by Onset Detection

In this section, we will explain how to segment an audio stream into events, using onset detection. In order to be more generally applicable, we have employed the complex domain based onset detector [10], since it subsumes onset detection algorithms based on energy, spectral difference, or phase as special cases. This onset detection function captures onsets due to abrupt energy changes as well as soft onsets induced by pitch changes, with little energy variations. For each frame l, the short-term Fourier transform yields a complex spectrum $X_k(l) = r_k(l)e^{i\phi_k(l)}$, with magnitude r_k and phase ϕ_k for the k-th bin with frame length K $(0 \le k \le K-1)$. We build the onset detection function as the Euclidean distance between the actual complex spectrum $X_k(l)$ at bin k and the estimated complex spectrum [10]:

$$\hat{X}_k(l) = \hat{r}_k(l)e^{i\hat{\phi}_k(l)}, \tag{1}$$

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 linear extrapolation from the unwrapped phases of the two preceding frames:

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

where the $\tilde{\varphi}$ denotes the unwrapped phase and the princarg operator maps the unwrapped value back to the $(-\pi, \pi]$ range. We calculate the bin-wise Euclidean distance between the actual and the estimated complex spectrum, quantifying the stationarity for the k-th bin as: $\Delta_k(l) = \|X_k(l) - \hat{X}_k(l)\|$. By summing across all K bins and across $M+1$ consecutive frames centered around frame l (smoothing), we yield the onset detection function:

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

Similarly to previous approaches [4], an adaptive threshold $\theta(l)$ is used. This threshold is calculated as the scaled median across a look-ahead window of length $P+1$

$$\theta(p) = C \cdot \text{median}_{n \in (p, p+1, \ldots, l+P)}(\eta(n)), \tag{3}$$

with $0 \le C \le 1$ being a predefined parameter controlling the sensitivity of the onset detector. In order to eliminate multiple occurrences of onsets shortly one after another, smoothing is applied via another window of length $W+1$ centered at sample l:

$$\mu(l) = \sum_{m=\lceil -\frac{W}{2} \rceil}^{\lceil \frac{W}{2} \rceil} \max(\eta(l+m) - \theta(l+m), 0) \tag{4}$$

A silence threshold θ_s is applied:

$$\mu_s(l) = \max(\mu(l) - \theta_s, 0). \tag{5}$$

Finally, the local maxima of $\mu_s(l)$ define the predicted onset times.

B. Feature Extraction for Timbre Representation

For each onset, a short window of length L subsequent to the onset time is analyzed. For each frame within this window, the first 13 Mel-Frequency Cepstrum Coefficients (MFCC) [31] are calculated. To model the coefficient's temporal behaviour right after the onset, for each coefficient another Discrete Cosine Transform (DCT) is calculated on the sequence of coefficients across the frames. Taking the first 4 DCT coefficients for each MFCC yields a 52-dimensional vector, representing timbral features both of the sound event's spectral characteristics and their initial temporal development.

C. Incremental Clustering for Symbol Sequence Generation

The clustering stage receives multivariate feature vectors from the preprocessing stage and converts them into symbols. 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 of future events. As a reference and benchmark, we compare online clustering by Cobweb with a state-of-the-art batch clustering method exploiting sequential information, the HDP-HMM.

1) Cobweb: For this purpose, Marxer et al. [28] used the Cobweb [12]. 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 modelled by a univariate Gaussian for each feature dimension. The edges of the structure represent taxonomic relations. Further works [29, 45] 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 [16]. The version of Cobweb that we will use was presented as Cobweb/3 [29] and later extended as Cobweb/95 [45]. This algorithm clusters D-dimensional feature vectors $\mathbf{x} = (x_1, \ldots, x_D)$ extracted in the previous section. Consider a particular cluster containing I feature vectors. Let σ_d be the standard deviation in component d of the input feature vectors assigned to that cluster. Then $\sum_{d=1}^{D} \frac{1}{\sigma_d}$ is the specificity of that cluster across all feature dimensions. We consider the utility U to quantify the gain in specificity by splitting this cluster into K child clusters. For a potential child cluster $1 \leq k \leq K$ with I_k instances and each input feature dimension d we define σ_{dk} to be the inner cluster standard deviation in that dimension. Then $\sum_{d=1}^{D} \frac{1}{\sigma_{dk}}$ is the specificity of cluster k, and $\sum_{k=1}^{K} \frac{I_k}{I} \sum_{d=1}^{D} \frac{1}{\sigma_{dk}}$ is the specificity of the child clusters altogether. For the cluster utility holds

$$U \propto \frac{1}{K} \left(\sum_{k=1}^{K} \frac{I_k}{I} \sum_{d=1}^{D} \frac{1}{\max(\sigma_{dk}, a)} - \sum_{d=1}^{D} \frac{1}{\sigma_d} \right), \qquad (6)$$

The acuity parameter a is an upper limit of maximal specificity (minimal standard deviation) of the clusters, thereby controlling the maximal resolution of the clustering discrimination.

The incorporation of an object is a process of clustering 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 clusters 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 clustering tree. As a search strategy we use hill-climbing through a space of clustering trees.

Thereby, the input is converted into a sequence not only of symbols, but also of meta symbols (partitions) according to their parent nodes and grandparent 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*,
- removing a partition, reparenting it's children if it has any.

2) Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM): The feature vector sequence may also be modelled as the emission of a HDP-HMM, a Bayesian nonparametric model in which the hidden states can be considered as clusters. Given the observed feature vector sequence, the most likely hidden state sequence can be interpreted as a sequence of symbols. In the HDP-HMM, the hidden states are assumed to be drawn from a countably infinite state space. The HDP-HMM is used to jointly estimate the number of clusters, the cluster assignment of the feature vectors, and the transition probabilities between clusters. Inference in the HDP-HMM is performed using the weak limit approximation [13] implemented in `pyhsmm` [19].[1] However the inference does not work in an online manner, it requires the entire feature vector sequence as input. This method is offline (batch mode) and is only used as a reference and benchmark, since it does not fulfil the constraint of clustering the feature vectors as they arrive to perform immediate prediction from the very beginning.

D. Sequence Analysis for Next Symbol/Onset Prediction

We choose two methods (hierarchical N-grams and conceptual Boltzmann machine) [37] that require relatively little storage by deducing frequency counts for longer sequences from frequency counts of their shorter subsequences. These algorithms iteratively predict the next symbol (or the inter-onset interval= IOI respectively) c_{t+1} based on previous symbols (IOIs) $c_{t-n+1}, c_{t-n+2}, \ldots, c_t$, previously generated by incremental clustering. Thereby we derive which sound to expect when. The prediction of symbols and of IOIs is performed independently. By predicting the IOI, we can determine the onset time of symbol c_{t+1}.

[1]http://github.com/mattjj/pyhsmm

1) Hierarchical N-Grams (HN): N-grams have been used in the analysis of genome sequences and in language modeling [48]. Exhaustive N-grams count the instances of all possible symbol (IOI) sequences of length N. Their memory requirement is exponential in the sequence length N and the problem arises how to account for patterns that have not occurred before (zero frequency problem). We use N-grams as an estimate for the forward conditional distribution for online prediction of the next symbol (IOI).

Hierarchical N-grams (HN) [37] need less memory than exhaustive N-grams. HN are a combination 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 statistics 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.

Let $\mathcal{C}_1 = \{c^1, \ldots, c^{|\mathcal{C}_1|}\}$ be 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 clustering process in the previous section. $\mathcal{C}_1$ forms the alphabet of the n-gram. Then, $\mathcal{C}_n$ is the set of all possible n-grams of length n composed from alphabet $\mathcal{C}_1$. To exploit sparsity, we only consider the patterns that have actually occurred as a subsequence of $\mathbf{c}$ so far until time t. The set of patterns of length n having occurred so far will be denoted by $\mathcal{C}_n = \{\mathbf{c}^1, \ldots \mathbf{c}^{|\mathcal{C}_n|}\}$, in which again the subpatterns are ordered according to their first appearance. $o(\mathbf{c})$ is defined as the position of $\mathbf{c}$ in $\mathcal{C}_{|\mathbf{c}|}$. We consider HNs of maximal length N. Let $C_{n,i} (n \leq N)$ be the frequency count of the i-th pattern of length n and let $T_{n,i}$ be the total count of patterns of length n since pattern i occurred for the first time. In Algorithm 1, we use the counts $T_{n,i}$ and $C_{n,i}$ to iteratively estimate the joint probabilities $P_{n,i}$ for all patterns seen so far. We define $T_{n,0} := T_{1,1}$ for $1 \leq n \leq N$. In simple N-grams, the empirical frequency $\frac{C_{n,i}}{T_{n,1}}$ could be used as an estimate for the probability of a pattern of length n. In the HN method (Eq. 12), the probability $P^{n-1}_{n,i}$ for the i-th pattern of length n under the joint distribution of width $n-1$ is estimated. For pattern $\mathbf{c}^i = (c^i_1, \ldots, c^i_n)$, statistical estimates (Eq. 8.2 in Pfleger)

$$P^{n-1}_{n,i} = \frac{P(c^i_1, \ldots, c^i_{n-1}) \cdot P(c^i_2, \ldots, c^i_n)}{\sum_{y \in \mathcal{C}_1} P(c^i_2, \ldots, c^i_{n-1}, y)} \tag{7}$$

are calculated, using the sub-patterns of the ith pattern of lengths n. We estimate the probability $Q^{n-1}_{n,i}$ (Eq. 11) of sub-patterns of length $n-1$ of the ith pattern of length n of not being a subpattern of the first i patterns of length n. In Eq. 12, they are weighted by their confidence. The confidence values depend on the number of occurrences of the 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 is not reliable. In this case the probability of appearance is better estimated from the $n-1$ length sub-patterns through $P^{n-1}_{n,i}$. In other words, the information of patterns of large lengths is integrated with the information of models of small lengths. Pfleger shows that the probability of a given pattern can be calculated in a linear sweep by updating all the probabilities in order of the pattern's first occurrence and length.

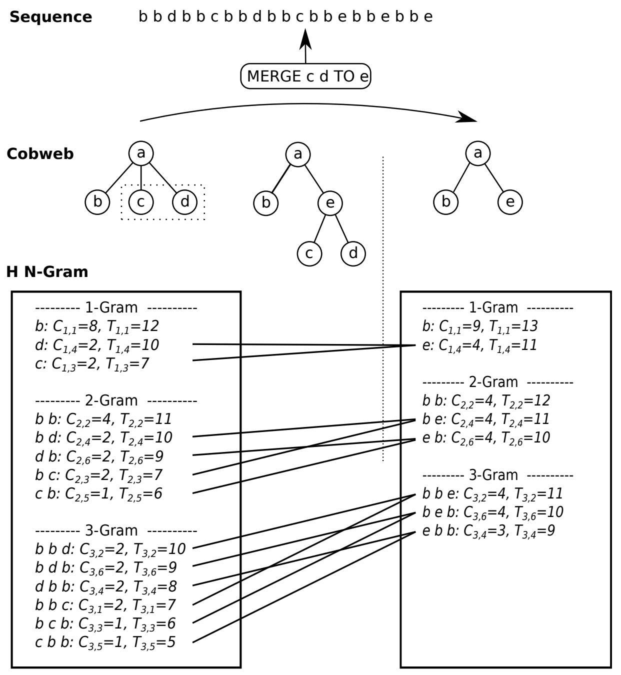
Sequence
b b d b b c b b d b b c b b e b b e b b e
MERGE c d TO e
Cobweb
a
b
c
d
e
H N-Gram
--------- 1-Gram ----------
b: $C_{1,1}$=8, $T_{1,1}$=12
d: $C_{1,4}$=2, $T_{1,4}$=10
c: $C_{1,3}$=2, $T_{1,3}$=7
--------- 2-Gram ----------
b b: $C_{2,2}$=4, $T_{2,2}$=11
b d: $C_{2,4}$=2, $T_{2,4}$=10
d b: $C_{2,6}$=2, $T_{2,6}$=9
b c: $C_{2,3}$=2, $T_{2,3}$=7
c b: $C_{2,5}$=1, $T_{2,5}$=6
--------- 3-Gram ----------
b b d: $C_{3,2}$=2, $T_{3,2}$=10
b d b: $C_{3,6}$=2, $T_{3,6}$=9
d b b: $C_{3,4}$=2, $T_{3,4}$=8
b b c: $C_{3,1}$=2, $T_{3,1}$=7
b c b: $C_{3,3}$=1, $T_{3,3}$=6
c b b: $C_{3,5}$=1, $T_{3,5}$=5
--------- 1-Gram ----------
b: $C_{1,1}$=9, $T_{1,1}$=13
e: $C_{1,4}$=4, $T_{1,4}$=11
--------- 2-Gram ----------
b b: $C_{2,2}$=4, $T_{2,2}$=12
b e: $C_{2,4}$=4, $T_{2,4}$=11
e b: $C_{2,6}$=4, $T_{2,6}$=10
--------- 3-Gram ----------
b b e: $C_{3,2}$=4, $T_{3,2}$=11
b e b: $C_{3,6}$=4, $T_{3,6}$=10
e b b: $C_{3,4}$=3, $T_{3,4}$=9

Fig. 2. 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.

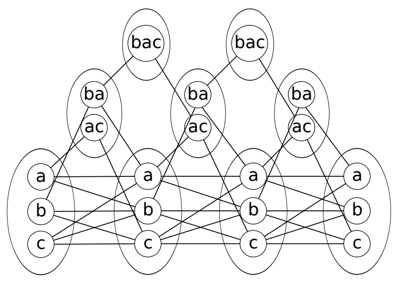

Fig. 3. Illustration of the continuous composition of symbols (atoms) in the Boltzmann Machine into longer patterns (chunks).

In order to adapt Pfleger's HN [37] to our architecture, we have to link the operations of the clustering model to the operations on the n-gram (Fig. 2). When two or more clusters are merged in the clustering model, we have to remove the superfluous clusters from the set of cluster indices (Eq. 8) and to sum up the counts for the merged clusters (Eq. 9). 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 with the count of *bbe*.

2) Conceptual Boltzmann Machine (CB): The Boltzmann machine [1] is a stochastic, symmetric-recurrent neural network that can be used to represent a joint distribution of

Algorithm 1 The Hierarchical N-Gram for Merged Clusters

```
Initialization $\mathcal{C}_n = \{\}$ for $1 \leq n \leq N$
for incoming event $c_t$ do
  for $1 \leq n \leq N$ do
    if $(c_{t-n+1}, \dots, c_t) \notin \mathcal{C}_n$ then
      Add new pattern: $\mathcal{C}_n = \mathcal{C}_n \cup (c_{t-n+1}, \dots, c_t)$, $T_{n,|\mathcal{C}_n|} = 0, C_{n,|\mathcal{C}_n|} = 0$
    end if
    if $\mathbf{c}^1, \dots, \mathbf{c}^k \in \mathcal{C}_n$ are merged by Cobweb then
        $o' = \min(o(\mathbf{c}^1), \dots, o(\mathbf{c}^{\mathbf{k}}))$                    (8)
        $C_{n,o'} = \sum_{i=1}^{k} C_{n,o(\mathbf{c}^k)}$                    (9)
        $\mathcal{C}_n = \mathcal{C}_n \backslash \{\mathbf{c}^1, \dots, \mathbf{c}^{o'-1}, \mathbf{c}^{o'+1}, \dots, \mathbf{c}^k\}$                    (10)
      Update indices
    end if
    Update counts: $C_{n,o(c_{t-n+1},\dots,c_t)} = C_{n,o(c_{t-n+1},\dots,c_t)} + 1$
    Update total counts: $T_{n,i} = T_{n,i} + 1$ for $1 \leq i \leq |\mathcal{C}_n|$
  end for
  Calculate joint probabilities:
  $P^0_{1,i} = \frac{1}{|\mathcal{C}_1|}$ $(1 \leq i \leq |\mathcal{C}_1|)$
  for $1 \leq n \leq N$ do
    for $1 \leq i \leq |\mathcal{C}_n|$ do
        $Q^m_{n,i} = (1 - \sum_{k=1}^{i} P^m_{n,k})$    $(m = n, n-1)$                    (11)
      Calculate $P^{n-1}_{n,i}$ according to Eq.7
        $P^n_{n,i} = \frac{1}{T_{1,1}} \left[ C_{n,i} + \sum_{j=0}^{i-1} (T_{n,j} - T_{n,j+1}) \cdot Q^n_{n,j} \cdot \frac{P^{n-1}_{n,i}}{Q^{n-1}_{n,j}} \right]$                    (12)
    end for
  end for
end for
```

random variables, to complete patterns, and in particular (as in our case) to predict the continuation of a time series.

Formally, a Boltzmann machine consists of a vector of binary units $(s_1, \dots, s_I) \in \{0,1\}^I$, and symmetric weights $w_{ij} \in \mathbb{R}$ between pairs of units (s_i, s_j), an update rule for the units and a learning rule for the weights.

Applied to *categorical* data [38], a V-valued symbol c_u is encoded as binary units $s_{u_1}, \dots, s_{u_V}$ with $s_{u_v} = 1$ if and only if $c_u = v$. To connect two V-valued variables c_u and c_i, V^2 weights w_{i_j,u_v} are needed to connect the binary units representing the two variables. Initially, the architecture of our particular Boltzmann machine implementation consists of sets of binary variables for consecutive symbols, where the binary nodes of each variable are initially only connected to the binary nodes of the previous and the next symbol. Depending on the other units and weights, the stochastic softmax update rule for the symbol is:

$$P(c_i = j) = \frac{1}{1 + e^{-\frac{\sum_{u \neq i} \sum_{v=1}^{V} s_{u_v} w_{i_j,u_v}}{T}}}, \tag{13}$$

with temperature T decreasing from $T = 50$ to $T = 0.005$ in 100 steps. As an example of Gibbs sampling, this update rule is applied iteratively. In general, through *simulated annealing* of the temperature T, the states converge to a particular state vector [1].

For training the Boltzmann machine, the weights w_{ij} have to be learned. As in the case of the restricted Boltzmann machine [41], in our case, not all pairs (s_i, s_j) are connected by non-zero weights w_{ij}. Units representing the same symbol are not connected among each other. For each binary previous symbol sequence, the update rule (13) is iteratively applied until the final states are reached (denoted by s_i^+). In addition, the update rule is applied with no units fixed until another vector of final states (s_i^-) is reached. Then a stochastic gradient-based learning step for the weights can be performed with learning rate μ for a single training instance yielding $s_i^+ s_j^+$:

$$\Delta w_{ij} = \mu(s_i^+ s_j^+ - s_i^- s_j^-). \tag{14}$$

The learning step aims at minimizing the difference between $s_i^+ s_j^+$ and $s_i^- s_j^-$. $\mu = 0.1$ is used.

When weight w_{ij} rises above a threshold θ_w, a new *hidden* unit is created, representing the concatenation of symbols connected by strong weights (cf. Fig. 3) [38]. In addition, weight w_{ij} is removed. Iteratively, hidden units for patterns of length $n+1$ are created from nodes representing patterns of length n and a new set of binary nodes representing patterns of length n is appended. We set $\theta_w = 0.2, 0.15, 0.1, 0.05$ respectively, depending on the length of the pattern the unit represents (length 1,2,3,4). This variant of the Boltzmann machine is called the *compositionally-constructive categorical Boltzmann machine* [38]. For predicting the next symbol c_{t+1}, in a trained Boltzmann machine, the respective units are fixed to the previous symbol sequence $c_{t-n+1}, \ldots, c_t$. After running the unit update rule (13) until convergence, the predicted next symbol c_{t+1} in the sequence can be retrieved from the corresponding binary units of the Boltzmann machine.

In our system we have implemented a new method called *conceptual Boltzmann machine (CB).* In Pfleger, the Boltzmann machine acts on a static set of categories. We have extended this to an architecture which operates on a dynamically changing taxonomy of categories. Therefore, the model adjusts to the tree structure generated by the Cobweb. This means the Boltzmann machine changes the architecture on the fly guided by the creation, removal, splitting, and merging operations suggested by the Cobweb. Accordingly, in the Boltzmann machine, the units and the update rule must be adjusted to the new structure.

During the run, sequences of atoms cause the creation of higher-level chunks that represent patterns. The newly created chunks that represent patterns are then further chunked into nodes that represent patterns of longer length. The longest pattern represented by a node is fixed to a value of N.

III. Performance Analysis of the System

A. Measures for Clustering Evaluation

Unlike in supervised learning, where accuracy can be measured between the annotated labels and the labels predicted by a classifier, the number of clusters predicted by the analysis can be different from the number of annotated label categories. In addition, the mapping between annotated and predicted labels is unclear. This creates the need for a particular clustering evaluation measure. The following measures for evaluating the agreement between annotations and predicted labels have been suggested: purity [47], F-measure [21], and Pearson's chi squared coefficient [44], and Rand index [44]. We choose the latter measure for evaluation, since it is a natural extension of classifying elements to pairs of elements.

A partition (clustering) $\mathcal{C}$ of a set $\mathcal{X}$ is defined as a set $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_J\}$ of subsets $\mathcal{C}_j \subset \mathcal{X}$, so that $\cup_j \mathcal{C}_j = \mathcal{X}$ and $\mathcal{C}_j, \mathcal{C}_{j'}$ disjoint for $j \neq j'$. Let $\mathbb{P}$ be the set of all partitions of $\mathcal{X}$ and let $|\mathcal{C}|$ be the number of elements in a partition $\mathcal{C} \in \mathbb{P}$. Let $\mathcal{A} \in \mathbb{P}$ be a partition generated by annotation and let $\mathcal{C} \in \mathbb{P}$ be a predicted partition derived from an algorithm. Let $\mathcal{P} = \{(x, x') | x, x' \in \mathcal{X}, x \neq x'\}$ be the set of pairs of distinct events. Let $\mathcal{L} \subset \mathcal{P}$ be the set of events pairs where both x and x' share the same labels/annotation provided by $\mathcal{A}$ and let $\mathcal{K} \subset \mathcal{P}$ be the set of event pairs where both x and x' lie in the same cluster provided by $\mathcal{C}$. Then $|\mathcal{K} \cap \mathcal{L}|$ are the number of point pairs that lie in the same cluster and - at the same time - share the same annotated labels. For $|\mathcal{C}| > 1$, the Rand index [44] is defined as:

$$R(\mathcal{A}, \mathcal{C}) = \frac{2(|\mathcal{L} \cap \mathcal{K}| + |\mathcal{P} \backslash \mathcal{L} \cap \mathcal{P} \backslash \mathcal{K}|)}{|\mathcal{C}|(|\mathcal{C}| - 1)}. \tag{15}$$

Since R depends on the number of clusters $|\mathcal{C}|$, we adjust the Rand index, comparing it with the expected value of R (baseline of a random clustering) E_R. The expected value of R over all partition combinations $\mathcal{P} \times \mathcal{P}$ is calculated as [14]:

$$E_R = \frac{1}{|\mathcal{C}|^2} \sum_{\mathcal{A}, \mathcal{C} \in \mathbb{P}} R(\mathcal{A}, \mathcal{C}). \tag{16}$$

E_R gets maximal for $\mathcal{A} = \mathcal{C}$:

$$R_{\max} = \frac{1}{|\mathcal{C}|^2} \sum_{\mathcal{C} \in \mathcal{P}} R(\mathcal{C}, \mathcal{C}). \tag{17}$$

Then the *adjusted Rand index (ARI)* holds:

$$ARI(\mathcal{A}, \mathcal{C}) = \frac{R(\mathcal{A}, \mathcal{C}) - E_R}{R_{max} - E_R} \tag{18}$$

The ARI has values between 0 (random partitioning) and 1 ($\mathcal{A} = \mathcal{C}$).

The ARI assumes that annotations and clusterings are drawn randomly with a fixed number of clusters and a fixed number of elements per cluster [44]. Although this assumption will not always be true in our evaluation, we will use ARI, since it is a more established measure than alternative ones, such as the Fowlkes-Mallows index, the Mirkin metric, the Jaccard index [30], or entropy-based measures [44, 47], e.g. normalized mutual information and variation of information.

In evaluating our system, we use the ARI in two ways: in the evaluation of 1) the clustering of the feature vectors of each event (Tables IV and V) and of 2) the prediction of the entire symbol sequence, as explained in the sequel. According to Fig. 1, by segmentation, feature extraction, and clustering, the input sound wave is transformed into a sequence $\mathbf{c} = (c_1, c_2, \ldots, c_T)$ of T events, each one represented as one of J symbols. All occurrences of symbol j can be included in a cluster $\mathcal{C}_j$ that contains all the indices t where event c_t equals symbol j. Then $\mathcal{C} = (\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_J)$ is a partition of $\mathcal{X} = (1, 2, \ldots, T)$. To evaluate the prediction $\mathbf{c}$, we annotate one of the ground truth labels $(1, 2, \ldots, I)$ to each segment of the input, yielding an annotated sequence $\mathbf{a} = (a_1, a_2, \ldots, a_T)$. From this, a partition $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_I)$ can be generated in the same way as the partition $\mathcal{C}$ for $\mathbf{c}$. The number I of the annotated labels is not necessarily the same as the number of symbols J determined by the clustering stage of our system. Then the ARI can be used to compare $\mathcal{C}$ and $\mathcal{A}$, as done in Tables I-II and Fig. 4-6.

B. Data Sets

Two sets of test data are employed:

- Repetitive symbol sequences: We generate sequences that consist of patterns of length $n_l = 2, \ldots, 5$ made up of I distinct symbols. These patterns are repeated 20

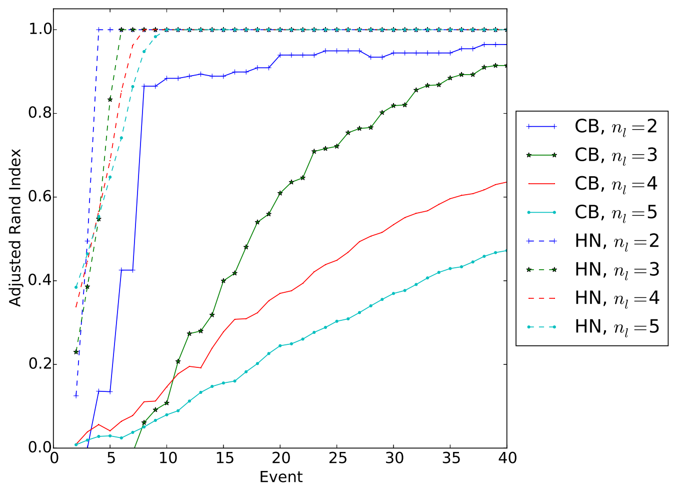

Fig. 4. Learning rate of the two sequence learning algorithms (CB and HN), depending on the number of pattern repetitions. The ARI (Eq. 18) is given for an increasing number of repetitions of a pattern with various lengths $n_l = 2, 3, 4, 5, 6$. HN reaches a perfect ARI quickly, in contrast to CB.

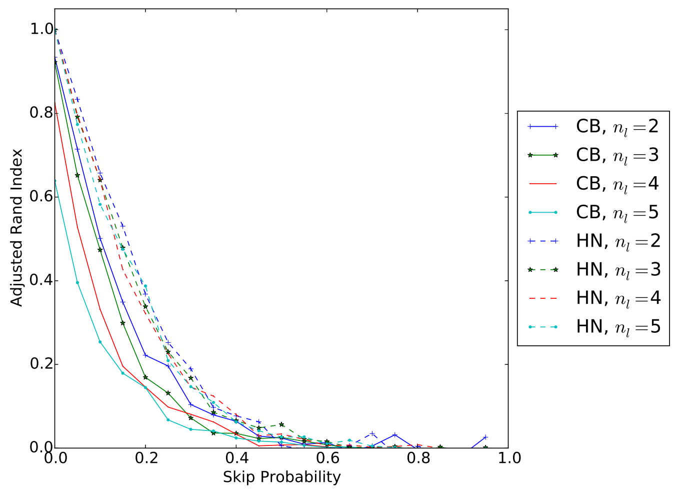

Fig. 5. Robustness of the two sequence learning algorithms (CB, HN, cf. Fig. 4) with respect to skipping noise. The ARI is given for a sequence of 20 repetitions of patterns of different lengths n_l and increasing probability p_{sk} of randomly skipping an event. For $p_{sk} < 0.4$, HN performs better than CB.

times. For each pattern length n_l and each partition $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_I\}$ of $(1, 2, \ldots, n_l)$, one sequence is generated in a way so that elements of each partition subset $\mathcal{A}_i$ are symbol i's positions in the sequence. E.g. for $n_l = 5$ and partition $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\} = \{\{1, 3, 5\}, \{2, 4\}\}$, symbol '1' occurs at positions $\mathcal{A}_1 = \{1, 3, 5\}$ and symbol '2' occurs at positions $\mathcal{A}_2 = \{2, 4\}$, yielding the symbol sequence ('1','2','1','2','1').

- Audio recordings:

Voice: Informal low quality and short voice recordings of very simplified beat boxing, each consisting of 2-3 different sound categories with different degrees of tonality with a simple changing rhythm, a sequence of a repetitive three-sound pattern, and a ritardando, altogether 5 recordings each of 10-13 s duration. In order to demonstrate the unsupervised character of our system we choose sounds that do not belong to a predefined category (e.g. an acoustical instrument).

ENST Drums: Formal high quality and automatically annotated recorded drum sequences. 5 segments described in terms of style, complexity and tempo as disco (simple slow, complex medium), rock (simple fast), country (simple slow, complex medium) [15].

The audio recordings are annotated, so they can be evaluated. Audio data is available on the website [27].

C. Results

The system architecture consists of the processing chain: 1) *onset detection and feature extraction* 2) *clustering*, 3) *expectation*. We will evaluate stages 1), 2), 3) in isolation, 1) + 2) together (referred to as *transcription*) and the entire chain 1) + 2) + 3) together (referred to as *prediction*). We use the repetitive symbol sequences, in order to assess expectation, i.e. learning rate and noise robustness of the sequence analysis. The audio recordings are used to test the processing stages of the entire system separately.

1) Learning Rate and Noise Robustness with Repetitive Symbol Sequences: We assess the learning rate and noise robustness of the sequence analysis stage (CB and HN). The sequence learning algorithm receives an initial chunk of a repetitive symbol sequence (Section III-B) as input. From this, the algorithm determines the most probable (expected) next $4n_l$ symbols. The algorithm outputs the expected next symbols $c_{t+1}, \ldots, c_{n_l \cdot 4}$, given the annotated symbols $a_1, a_2, \ldots, a_t$. For each t, from the predictions $c_{t+1}, c_{t+2}, \ldots, c_{n_l \cdot 4}$ a partition $\mathcal{C}$ is generated and compared with the partition of the corresponding $\mathcal{A}$ based on annotations, as explained in Section III-A. For $t \leq n_l \cdot 5$ all annotations so far are used for prediction, then only the last 12 annotations $a_{t-11}, a_{t-10}, \ldots a_t$ are used for prediction. For the stochastic BM, ARI is averaged over 100 runs of all partitions of a given length n_l. The trivial sequence that consists of a constant repetition of the same symbol is not considered. First we assess how the learning rate scales with *pattern length* and number of *pattern repetitions*. Fig. 4 shows the averaged ARI across of all partitions of lengths $n_l = 2, \ldots 5$. For this test, the HN is set to a maximum N-gram length of $N = 5$. The HN reaches perfect prediction (ARI=1) after $4n_l$ events (2 pattern repetitions). CB seems to converge much more slowly than HN, for $n_l = 2$ reaching an ARI of higher than 0.8 after 8 events, then increasing much more slowly. For higher n_l, CB seems to converge towards perfect prediction even more slowly.

Different types of noise are used to transform the sequence in order to assess the robustness of the sequence learning techniques:

Skipping noise: In the original sequence, a symbol is skipped with a given probability $0 \leq p_{sk} \leq 0.95$.

Switching noise: In the original sequence, with a given probability of $0 \leq p_{sw} \leq 0.95$, a symbol is selected randomly with uniform distribution across the n_l alternative symbols.

The average ARI is calculated over 100 runs for $n_l = 2, 3$, over 50 runs for $n_l = 4$ and 20 runs for $n_l = 5$ for both

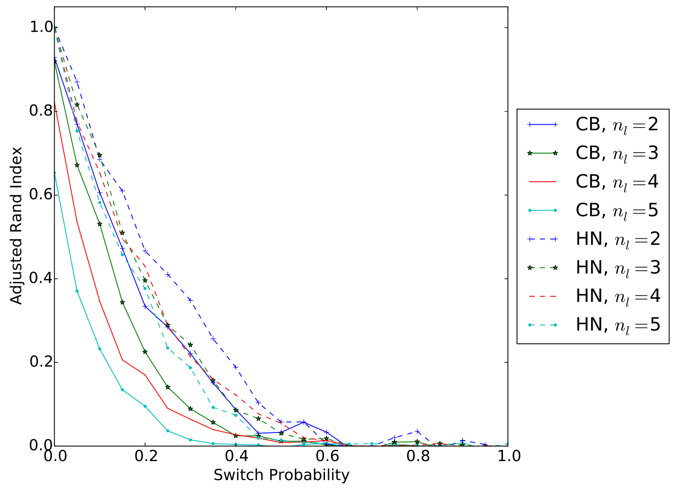

Fig. 6. Robustness of CB and HN (cf. Fig. 4) with respect to switching noise. The ARI is given for an increasing probability p_{sw} of randomly switching a symbol. HN performs better than CB for $p_{sw} < 0$, reaching random guess level (ARI=0) for $p_{sw} = 0.5$.

CB and HN. Fig. 5 shows how the prediction performance (ARI) is affected by skipping symbols with a defined p_{sk} in the repetitive symbol sequences. This simulates e.g. the failure of the onset extraction algorithm to detect an event. The prediction is performed, given a sequence of 20 repetitions of the basic pattern. For HN and CB, the performance degrades until $p_{sk} = 0.5$, where random guess level is reached (ARI=0). Until $p_{sk} = 0.4$, HN appears to be more robust towards skipping noise than CB, with CB having a worse ARI for higher n_l.

In Fig. 6, the effect of clustering errors on the sequence learning process is simulated. With increasing switching probability p_{sw}, a symbol is replaced by any of the n_l symbols under uniform distribution. The graph shows the prediction performance using the ARI for CB and HN for different pattern lengths. The results are similar as for skipping noise (Fig. 5): HN is more robust wrt noise than BM, reaching random guess level (ARI=0) for $p_{sw} = 0.5$. It can be summarized that for relatively small noise the HN appears to be more robust to skipping and switching noise, especially for longer pattern lengths.

2) Testing of Processing Stages with Audio Recordings: The tests with the *Voice* recordings (Section III-B) serve as a proof of concept of clustering with dynamically varying numbers of clusters. The *ENST* recordings are used for a more comprehensive quantitative evaluation of the system. We test each process stage separately. For the audio, the sample rate is $f_s = 44100$ Hz. For segmentation and feature extraction, the hop size is 128 samples, and the window size is 1024 samples.

a) Onset detection: For the evaluation of the onset detection (Section II-A) we employ a widely used procedure [9, 23]. Onset times manually annotated by subjects serve as references. The onsets estimated by the onset detection algorithm are then compared to the manually annotated onsets. 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 to be monophonic, the evaluation only permits a one-to-one mapping between estimated and annotated onsets.

Using the following onset detection parameters: smoothing length $M = 33$ in Eq. 2, sensitivity $C = 0.9$ and look-ahead window length $P = 10$ in Eq. 3, threshold window length $W = 11$ in Eq. 4, and silence threshold $\theta_s = 0.002$ in Eq. 5, onset detection yields an F-measure of $\sim 99\%$ for the *Voice* data set. Therefore, we focus on the clustering and the prediction stage. We also notice that for smoothing lengths $M > 33$ the system does not improve significantly. Large smoothing lengths reduce the temporal precision of onsets, which is important for good feature extraction, since most of the information about an event is located in the attack.

b) Clustering: We now compare the performance of our incremental online Cobweb clustering and benchmark offline (batch) HDP-HMM clustering with a constant yet inferred cluster number as a benchmark. 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. In order to assess the stability of the system, we tested it performing a grid search on the two most sensible parameters involved in the task and the algorithm. For Cobweb, we explore the analysis window length L (Section II-B) and the acuity a (Eq. 6). On the parameter grid, the window length/acuty pair with maximal ARI is determined, extending the parameter grid if the maximum lies on the grid border, with empirically set constant grid step sizes.

For *Voice*, Cobweb performance peaks at ARI=82.7% for $L = 150ms, a = 18.5$ on a parameter grid over $L = 50, 75, \ldots, 175; a = 15, 15.5, \ldots, 19$. For *ENST*, Cobweb performs best at 85.7% for $L = 50ms, a = 13.5$ on a parameter grid over $L = 25, 50, \ldots, 100; a = 13, 13.5, \ldots, 15$. (cf. Tables IV and V in the supplementary material[27]) This means that the timbre model and clustering process can successfully classify the audio events. We also notice that *Voice* needs a much longer analysis window than *ENST*. 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 (Section III-C2c).

For the HDP-HMM, we first reduce the feature vectors of the input to D dimensions by means of a PCA on the full sequence. The observation distributions used are Gaussian with parameters sampled i.i.d. from a normal inverse Wishart prior [19] with parameters $\mu_0 = \mathbf{0}, \kappa_0 = 0.4, \Lambda_0 = 0.001, \nu_0 = D + 2$.[2] The maximum number of states of the weak limit approximation inference is set to 10 and the number of Gibbs sampling iterations to 100. For *Voice*, HDP-HMM performance peaks at ARI=99.1% for $\gamma = 8.0, \alpha = 7.0, D = 2$ on a parameter grid over $\gamma, \alpha = 4.0, 5.0, \ldots, 11.0$ and $D = 2, 3$. For *ENST*, HDP-HMM performs best at ARI=84.0% for HDP concentration parameters $\gamma = 6.0, \alpha = 12.0$ [43] and $D = 2$ on a parameter grid over $\gamma, \alpha = 4.0, 5.0, \ldots, 13.0$ and $D = 2, 3$. Benchmark HDP-HMM performs better for *Voice* than Cobweb, whereas for *ENST*, Cobweb performs 1.7%

[2]Cf. Murphy [33], Section 9.2., p. 20 for the meaning of the parameters.

TABLE I
EXPECTATION OF *Voice* (LEFT) AND *ENST* (RIGHT): ARI (IN %) FOR DIFFERENT MAXIMUM LENGTHS N OF CB/HN (ROWS).

N	CB	HN
2	7.4	22.4
3	6.8	27.3
4	7.3	41.1
5	5.1	**50.9**
6	4.4	50.9
7	5.1	50.9

N	CB	HN
2	6.0	18.9
3	9.1	28.9
4	7.8	**43.2**
5	6.3	42.7
6	6.6	42.7
7	7.7	42.6

better than HDP-HMM. When comparing these results one has to keep in mind that HDP-HMM clustering has learned offline jointly a stable cluster number and the transition probabilities, exploiting sequential information whereas Cobweb has been trained online with an adaptive cluster number.

c) Transcription: The transcription test evaluates the subsystem composed of onset detection, feature extraction, and clustering. In contrast to the expectation test, the entire symbol (inter-onset interval) sequence $c_1, c_2, \ldots, c_t$ extracted from the clustering stage is always used from the beginning to predict the next symbol (inter-onset interval) c_{t+1}. The annotations $\mathbf{a}$ are not used for prediction, only for evaluation. The partitions generated from the detected events were compared with the partitions generated from the annotated labels using ARI. Online learning Cobweb with dynamically changing clustering numbers and offline learning HDP-HMM with a constant cluster number are compared. For Cobweb, *Voice* performs with $ARI = 81.3\%$ for $L = 150, a = 17$. On the *ENST* data set Cobweb yields $ARI = 76.3\%$ for $L = 50, a = 13.5$, using the same parameter grids as for the clustering (p. III-C2b). In comparison to the results for clustering, the ARI degrades a bit in particular for *ENST* due to wrongly estimated onsets. (cf. Tables VI and VII in the supplementary material [27]) HDP-HMM transcription performance for *Voice* peaks at ARI=98.8% for $\gamma = 8.0, \alpha = 12.0$ on a parameter grid over $\gamma, \alpha = 4.0, 5.0, \ldots, 13.0$. For *ENST*, HDP-HMM transcription performance peaks at ARI=76.2% for $\gamma = 5.0, \alpha = 8.0$ on the same parameter grid as for *Voice*. For *ENST*, HDP-HMM and Coweb are almost equal. Although for *Voice*, the ARI is much higher for the HDP-HMM benchmark than for Cobweb, we have to keep in mind that Cobweb learns online with changing cluster numbers over time whereas HDP-HMM is trained offline with a constant number of clusters.

d) Expectation: The expectation test evaluates the performance of the sequence learning module on the data sets. We predict the cluster label c_{t+1} of event $t+1$ based on the annotations from the start: $a_1, a_2, \ldots, a_t$. Results in Table I show that for the prediction of the sequences of the *Voice* and the *ENST* data set, HN ($ARI = 43.2\%$ for $N = 4$) works a lot better than CB, which yields an $ARI = 7.8\%$, just slightly better than random (0%). CB's low performance can be attributed to various factors: In general, many traditional recurrent networks are known to have a slow learning rate.[18] In particular, we have observed slow learning rate (Fig. 4) and low noise robustness (Fig. 5 & Fig. 6). Whereas for HN, the updates in frequency counts are getting smaller relative to the count so far (from 1 to 2 is a higher step relative to 1 than from 100 to 101 relative to 100), in CB the weights

TABLE II
FULL PREDICTION FOR THE *Voice* DATA SET USING HN: ARI (IN %) FOR DIFFERENT TEMPORAL ACUITIES a_t FROM EQ. 6 (ROWS) AND TIMBRAL ACUITIES a FROM EQ. 6 (COLUMNS).

$a_t \backslash a$	17	17.5	18	18.5	19	19.5	20
0.05	16.7	15.5	16.3	16.0	25.2	23.8	23.8
0.0625	15.9	15.3	16.2	16.0	25.2	24.0	24.0
0.075	14.4	14.3	15.2	16.7	**27.2**	25.8	25.8
0.0875	15.7	16.4	17.2	17.3	25.8	24.5	24.5

TABLE III
FULL PREDICTION FOR THE *ENST* DATA SET USING HN: ARI (IN %) FOR DIFFERENT TEMPORAL ACUITIES a_t FROM EQ. 6 (ROWS) AND TIMBRAL ACUITIES a FROM EQ. 6 (COLUMNS).

$a_t \backslash a$	18	18.5	19	19.5	20	20.5	21
0.075	33.1	33.8	33.0	32.6	36.2	34.7	33.3
0.0875	35.1	36.2	35.3	34.2	37.8	36.3	34.9
0.1	36.3	37.6	36.6	35.4	**39.2**	37.6	36.2
0.1125	35.9	37.2	36.2	35.0	38.8	37.2	35.8
0.125	34.6	35.9	34.9	33.7	37.5	35.9	34.5

are updated with a constant learning rate μ (Eq. 14). Weight updates are performed by stochastic gradient descent where each instance is only used once when it has just occurred. Although this is cognitively plausible if we assume that only a limited number of instances can be stored by the cognitive system, it comes with the price of diminished learning speed, compared to a system where the update is performed using a batch of instances. In addition, the architecture of the CB may be suboptimal w.r.t. the hidden nodes. Also, in the network, new hidden nodes are only generated one at a time, further limiting learning speed. Furthermore, the parameters θ_k for creating new hidden nodes are chosen heuristically and may be suboptimal for short sequences like the one presented. We can also see that for n-gram maximum lengths $n > 5$ for *Voice* ($n > 3$ for *ENST*) the result does not improve. For linguistic data, slower convergence and worse performance of CB relative to HN is also observed in Pfleger [37], pp. 80&133. In the sequel, we will only use HN.

e) Prediction: The prediction task consists in running the full system including HN as the sequence analyzer (Tables II and III). After the transcription of the events $c_1, \ldots, c_t$, the system predicts the next symbol and the timing of it (the next IOI) c_{t+1}. For evaluating the match between predicted and annotated onsets, we set the tolerance threshold to 150 ms. For the best configuration, the full prediction yields an ARI of 27.2% for *Voice* and an ARI of 39.2% for *ENST*. The performance is limited by the weakest performance of its components, in this case the sequence analysis.

D. Examples

In this section, we present a few examples (audio on the website [27]) of transcription and prediction using HN in order to demonstrate the performance, evolution and shortcomings of the system. From Hazan et al. [17], we adopt the procedure to optimally map the annotated symbols to the clusters found by the clustering algorithm. We calculate the matching matrix between the annotations ('score') and clusters of each event. In this matching matrix, we then iteratively yield the maximal entry, thereby establishing a connection between a row (annotations) and a column (clusters). After eliminating the row and column of the maximal entry, we determine the maximal entry again until the matrix vanishes.

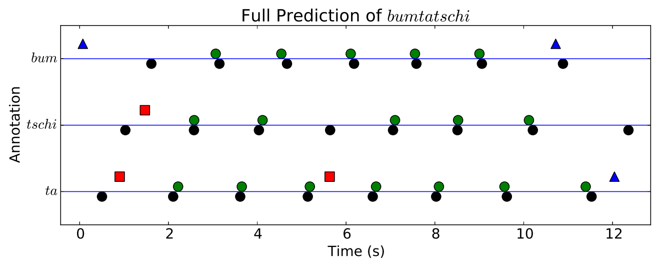

Fig. 7. The system (with HN) quickly captures a simple *ta-tschi-bum* pattern. Time (horizontal axis) is mapped versus event labels (one line each for *'ta', 'tschi',* and *'bum'*). Annotated labels are indicated in black below the lines. Above the horizontal lines we find events that are correctly estimated ('●'), matched to the wrong cluster ('■'), and unmatched ('▲') due to a wrongly estimated onset.

In Fig. 7 and 8, we display sequences of annotations and clusters on the same line if they are linked through this mapping. In Fig. 7, a simple *ta-tschi-bum* pattern is quickly captured. We can see how the first three events annotated as *ta*, *tschi*, and *bum* are matched with the wrong clusters *bum* (initial blue triangle above top line), *ta*, and *tschi* (red squares). The first three cluster mismatches are expected, since the system has no previous knowledge of the symbol space nor of the sequence and therefore cannot predict symbols nor patterns that have not yet occurred. At around $5.5s$, an event annotated as *tschi* is matched with the *ta* cluster. The timing of the last *bum* is misestimated and for the last *tschi* timing and cluster matching are wrong. The time deviation errors are due to the fact that the recorded voice does not follow a temporally regular pattern.

In Fig. 8, we observe how the system adapts to pattern changes within the sequence. For the first two events, the cluster matching is wrong. Then, after having processed enough sounds, the system performs correct predictions. In the middle, around $7.5s$, when the repetition pattern of *ta* is introduced, for three *ta* events the onset is wrongly estimated, two of these events as well being mismatched with the wrong cluster, and one additional event being only mismatched with the wrong cluster. The errors in the middle of the sequence are due to the pattern change. The N-gram is able to update the statistics

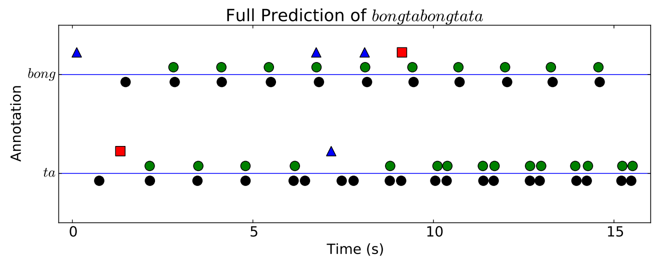

Fig. 8. The system (with HN) adapts to a pattern change from *ta-bong* to *ta-ta-bong* (cf. Fig. 7).

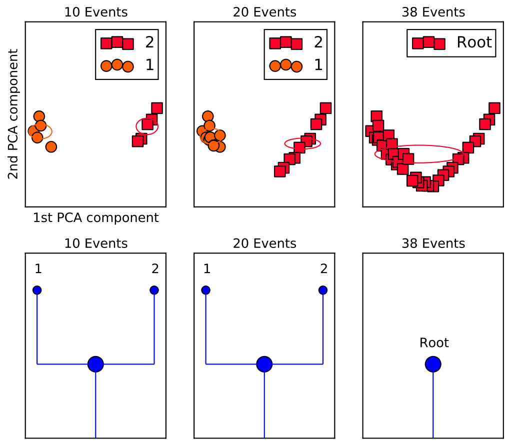

Fig. 9. Cluster merging: After 38 events, two clusters ('●', '■') merge into one cluster ('■'). The projection of the MFCC vectors (timbre representation) onto their first two principal components (*above*) and the incremental clustering tree (*below*) are shown.

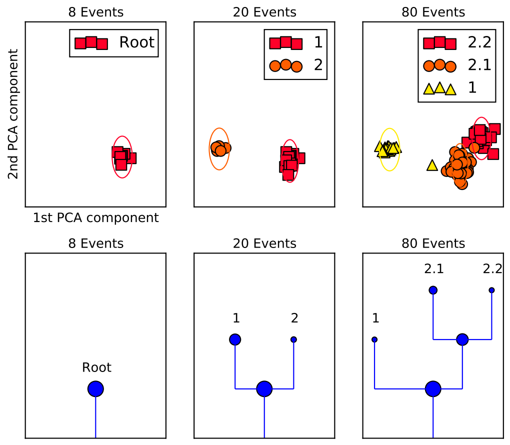

Fig. 10. Creation of new clusters: After 20 and 80 sound events, new clusters ('●', '▲') emerge on the fly. Cf. Fig. 9.

and perform correct predictions after three occurrences of the new pattern.

In Fig. 9 (sound and video on the website [27]), a sequence of alternating bass drum and hi-hat samples is played. During the sequence, the hi-hat is gradually mixed in a linear fashion with an increasing amount of bass drum and vice versa so that in the end both hi-hat and bass drum are mixed together in a balanced way, yielding a repetitive sequence of similar sounds. The system recognizes the two sound clusters in the beginning, and finally merges the two clusters into one single cluster.

In Fig. 10 (sound and video on the website [27]), a sequence of sound events is analyzed that starts with one sound, later joined by a second and third sound. The system is able to split the initial cluster gradually into 2 and 3 clusters.

IV. Conclusion and Perspectives

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 to pattern changes in the sequence as well as the appearance of new sounds or instruments at any time. Currently the system is limited by the lack of metrical analysis, making it especially sensitive to missed onsets. Considering the metrical context could significantly improve the quality of predictions. For this goal, a metrical alignment procedure[24, 25] could be combined with incremental learning. As alternatives to CB and HN, variable length Markov models [6, 24, 25] or other deep learning architectures can be used, thereby overcoming the context length limitation of HN and CB and the slow learning of CB. The long short-term memory (LSTM) [18] is a recurrent neural network that had been developed to capture dependencies between disconnected distant chunks within the same time series. Crucial to this and for speeding up learning, in LSTM, special memory cells are used. The access to the latter can be opened and closed by special gating units. Successfully applied to protein homology detection, automatic composition [11], handwriting and spoken language recognition, LSTM could be used to replace CB or HN and improve learning speed in our application. HDP-HMM [43] could be adapted to online learning [3] with incremental addition/removal of clusters comprising also segmentation, thereby replacing onset detection. The presented system can also be modified to learn melodies and chord progressions. For learning melodies, in the feature extraction stage (Section II-B), MFCCs need to be replaced by a pitch detection method as used in Marxer et al. [28] for learning songs by the Mbenzele pygmies or as in Cherla et al. [7] for learning guitar riffs. When analysing (piano) chord progressions, MFCCs can be replaced by constant Q profiles [20, 39]. Future work includes the development of a better representation of pitch and harmony, using a larger training set when processing more complex music.

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 its human partner, they could develop an enhanced musical conversation.

References

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 9(1):147–169, 1985.

[2] Gérard Assayag and Shlomo Dubnov. Using factor oracles for machine improvisation. *Soft Computing*, 8 (9):604–610, 2004.

[3] A. Bargi, R.Y.D. Xu, and M. Piccardi. An online HDP-HMM for joint action segmentation and classification in motion capture data. In *Computer Vision and Pattern Recognition Workshops, IEEE Computer Society Conf. on*, pages 1–7, June 2012.

[4] Juan Pablo Bello and Mark B. Sandler. Phase-based note onset detection for music signals. *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 5(2):441–444, 2003.

[5] Thierry Bertin-Mahieux, Ron J Weiss, and Daniel PW Ellis. Clustering beat-chroma patterns in a large music database. In *Proc. Int. Society Music Information Retrieval*, pages 111–116, 2010.

[6] Peter Buhlmann and Abraham J. Wyner. Variable length Markov chains. *Annals of Statistics*, 27:480–513, 1999.

[7] Srikanth Cherla, Hendrik Purwins, and Marco Marchini. Automatic phrase continuation from guitar and bass guitar melodies. *Computer Music Journal*, 37(3):68–81, 2013.

[8] Darrell Conklin and Ian H Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73, 1995.

[9] J. Stephen Downie, Kris West, Andreas F. Ehmann, and Emmanuel Vincent. The 2005 music information retrieval evaluation exchange (MIREX 2005): Preliminary overview. In *Proc. Int. Society Music Information Retrieval*, pages 320–323, 2005.

[10] C. Duxbury, J. Bello, M. Davies, and M. Sandler. Complex domain onset detection for musical signals. In *Proc. Digital Audio Effects Workshop*, London, UK, 2003.

[11] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In *IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756. IEEE, 2002.

[12] Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139–172, 1987.

[13] Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. A sticky HDP-HMM with application to speaker diarization. *The Annals of Applied Statistics*, pages 1020–1056, 2011.

[14] Ana LN Fred and Anil K Jain. Combining multiple clusterings using evidence accumulation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(6): 835–850, 2005.

[15] O. Gillet and G. Richard. Automatic transcription of drum loops. *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 4, 2004.

[16] M. Gluck and J. Corter. Information, uncertainty, and the utility of categories. *Proc. Annual Conf. of the Cognitive Science Society*, pages 283–287, 1985.

[17] 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, 2009.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Matthew J. Johnson and Alan S. Willsky. Bayesian nonparametric hidden semi-Markov models. *Journal of Machine Learning Research*, 14:673–701, February

2013.

[20] Katerina Kosta, Marco Marchini, and Hendrik Purwins. Unsupervised chord-sequence generation from an audio example. In *ISMIR*, pages 481–486, 2012.

[21] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proc. ACM SIGKDD Int. Conf. on knowledge discovery and data mining*, pages 16–22, 1999.

[22] Olivier Lartillot, Shlomo Dubnov, Gérard Assayag, and Gill Bejerano. Automatic modeling of musical style. In *Proc. Int. Computer Music Conference*, 2001.

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

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

[25] Marco Marchini and Hendrik Purwins. Unsupervised analysis and generation of audio percussion sequences. In *Exploring Music Contents*, pages 205–218. Springer, 2011.

[26] Ricard Marxer and Hendrik Purwins. Unsupervised incremental learning and prediction of audio signals. In *Proc. Int. Symposium on Music Acoustics*, 2010.

[27] Ricard Marxer and Hendrik Purwins. Unsupervised incremental learning and prediction of audio signals: Supplementary material, December 2014. URL http://www.ricardmarxer.com/research/unsupervised2014.

[28] Ricard Marxer, Piotr Holonowicz, Hendrik Purwins, and Amaury Hazan. Dynamical hierarchical self-organization of harmonic, motivic, and pitch categories. In *Music, Brain and Cognition Workshop, held at Neural Information Processing Conference*. 2007.

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

[30] Marina Meila. Comparing clusterings an information based distance. *Journal of Multivariate Analysis*, 98(5): 873 – 895, 2007.

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

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

[33] Kevin P. Murphy. Conjugate bayesian analysis of the gaussian distribution. Technical report, University of British Columbia, 2007.

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

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

[36] 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.

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

[38] Karl Pfleger. On-line learning of predictive compositional hierarchies by Hebbian chunking. Technical report, In Proceedings of the AAAI2000 workshop on New Research Problems for Machine Learning, 2002.

[39] H. Purwins, B. Blankertz, and K. Obermayer. A new method for tracking modulations in tonal music in audio data format. In *Int. Joint Conf. on Neural Network (IJCNN'00)*, volume 6, pages 270–275. NI-BIT, 2000.

[40] Lu Ren, David B Dunson, and Lawrence Carin. The dynamic hierarchical dirichlet process. In *Proc. Int. Conf. on Machine learning*, pages 824–831. ACM, 2008.

[41] Paul Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In J. McClelland D. Rumelhart, editor, *Parallel Distributed Processing*, volume 1, pages 194–281. MIT Press, Cambridge, MA, 1986.

[42] Thomas S Stepleton, Zoubin Ghahramani, Geoffrey J Gordon, and Tai S Lee. The block diagonal infinite hidden Markov model. In *Int. Conf. on Artificial Intelligence and Statistics*, pages 552–559, 2009.

[43] Yee Whye Teh and Michael I Jordan. Hierarchical Bayesian nonparametric models with applications. *Journal of the American Statistical Association*, 1, 2010.

[44] Silke Wagner and Dorothea Wagner. *Comparing clusterings: an overview*. Universität Karlsruhe, Fakultät für Informatik Karlsruhe, 2007.

[45] Jungsoon Yoo and Sung Yoo. Concept formation in numeric domains. In *Proc. ACM Annual Conf. on Computer Science*, pages 36–41, 1995.

[46] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. A probabilistic model for online document clustering with application to novelty detection. In *Advances in Neural Information Processing Systems 17*, pages 1617–1624. MIT Press, Cambridge, MA, 2005.

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

[48] Imed Zitouni, Olivier Siohan, Hong-Kwang Jeff Kuo, and Chin-Hui Lee. Backoff hierarchical class n-gram language modelling for automatic speech recognition systems. In *Proc. Interspeech*, 2002.

V. Supplement Results: Grid Search on Parameters

TABLE IV
Cobweb clustering of *Voice* data: ARI (in %) for different timbral acuities a from Eq. 6 (rows) and analysis window lengths L (Section II-B, columns) .

$L\backslash a$	15	15.5	16	16.5	17	17.5	18	18.5	19
50	31.0	33.0	33.0	33.0	33.0	34.3	35.3	35.3	35.3
75	62.1	57.8	42.5	42.2	46.2	46.2	34.6	34.6	36.2
100	76.5	77.6	75.8	78.6	81.0	78.9	55.7	55.7	37.9
125	79.8	79.8	71.5	73.3	73.6	73.7	78.0	78.0	80.6
150	67.5	66.6	67.7	68.6	73.3	76.2	81.2	**82.7**	78.6
175	60.0	65.6	67.2	67.4	70.1	73.5	74.2	75.5	72.4

TABLE V
Cobweb clustering of *ENST* data: ARI (in %) for different timbral acuities a from Eq. 6 (rows) and analysis window lengths (columns).

$L\backslash a$	13	13.5	14	14.5	15
25	65.6	64.7	63.9	63.9	62.8
50	83.0	**85.7**	80.5	80.2	76.7
75	73.9	72.5	69.6	78.3	65.8
100	74.2	69.1	67.9	66.3	68.8

TABLE VI
Onset and Cobweb transcription: ARI (in %) of acuity a from Eq. 6 for timbre clustering (rows) versus analysis window length L (columns) measured on the *Voice* data set.

$L\backslash a$	15	15.5	16	16.5	17	17.5	18	18.5	19
50	29.8	29.2	28.9	30.2	37.4	37.4	37.4	37.4	37.4
75	56.3	62.5	57.1	43.8	30.7	29.3	29.3	28.5	35.2
100	70.5	71.7	77.9	76.8	77.7	67.0	49.4	36.9	35.6
125	65.1	68.3	72.7	72.7	73.0	73.4	73.4	75.6	63.4
150	69.0	69.7	71.8	79.4	**81.3**	80.7	78.8	77.7	79.0
175	66.4	67.7	68.4	69.8	70.8	73.4	73.4	74.6	75.6

TABLE VII
Onset and Cobweb transcription: ARI (in %) of acuity a from Eq. 6 for timbre clustering (rows) versus analysis window length L (columns) measured on the *ENST* data set.

$L\backslash a$	13	13.5	14	14.5	15
25	65.9	67.1	67.1	67.1	67.0
50	67.5	**76.3**	71.4	71.4	71.4
75	63.2	61.4	60.6	70.8	70.2
100	62.4	57.2	57.9	58.1	61.6