EXPECTATION ALONG THE BEAT: A USE CASE FOR MUSIC EXPECTATION MODELS

Amaury Hazan, Paul Brossier, Piotr Holonowicz, Perfecto Herrera and Hendrik Purwins
Pompeu Fabra University
Music Technology Group
{ahazan, pbrossier, pholonow, pherrera, hpurwins}@iua.upf.edu

ABSTRACT

We present a system to produce expectations based on the observation of a rhythmic music signals at a constant tempo. The algorithms we use are causal, in order be fit closer to cognitive constraints and allow a future real-time implementation. In a first step, an acoustic front-end based on the *aubio* library extracts onsets and beats from the incoming signal. The extracted onsets are then encoded in a symbolic way using an unsupervised scheme: each hit is assigned a timbre cluster based on its timbre features, while its inter-onset interval regarding the previous hit is computed as a proportion of the extracted tempo period and assigned an inter-onset interval cluster. In a later step, the representation of each hit is sent to an expectation module, which learns the statistics of the symbolic sequence. Hence, at each musical hit, the system produces both *what* and *when* expectations regarding the next musical hit. For evaluating our system, we consider a *weighted average F-measure*, that takes into account the uncertainty associated with the unsupervised encoding of the musical sequence. We then present a preliminary experiment involving generated musical material and propose a roadmap in the context of this novel application field.

1. INTRODUCTION

We design and implement a musical learning system aimed at musical expectation and interaction, following [2, 19, 24, 27]. We emphasise the expectation of the attended musical audio, rather than other computer music applications such as improvisation or accompaniment, because it provides a way to evaluate models of musical expectation and memory. We first present some related work in the field of learning of musical sequences and musical expectation.

Sequence learning and structure acquisition have been modelled in a range of computational architectures. One approach to address this task consists in using connectionist architectures [28]. Among these methods, recurrent neural networks (RNN) can successfully deal with sequences of symbols [11, 17]. This is due to their ability to encode the context of the events, and the fact that the size of the context is not fixed. This is why RNN have been considered to address musical tasks. For instance, Mozer [23] applies them to melody and chord expectation tasks and shows how to learn musical structures in an invariant form. More recently, Eck and Schmidhuber [10] use the LSTM architecture [17] to learn Blues improvisations. On the other hand, Markov-chain models such as N-grams can be considered.

Such techniques have been long considered in musical applications from machine improvisation [19] to cognitive modelling of music perception [12]. A detailed review of monophonic musical sequence modelling is given in [26]. Other approaches make use of Markovian modelling to learn the structure of musical sequences in an interaction setting, the best known being the Pachet's Continuator [24]. Assayag and Dubnov [2] use a new pattern matching algorithm called the oracle factor [1] to hierarchically encode the presented sequences. This latter work has been applied to audio signals by using an acoustic front end. Nevertheless, little attention has been directed to non-symbolic approaches to musical sequence modelling. Raphael [27] investigates accompaniment and score following from audio signals using hidden Markov models and polyphonic transcription.

Also, Cemgil [6] uses dynamic Bayesian networks to perform rhythm and tempo tracking and polyphonic pitch tracking. Among the presented works, few have focused on complex polyphonic musical signals such as those found in commercial recordings. Schwarz [29] proposes a concatenative synthesis based on unsupervised techniques which is applied to music and speech. Jehan [18] considers learning and prediction applied to a wider range of music signals. The authors suggests several computational approaches to prediction of musical features (e.g. downbeat prediction), but do not consider the prediction of a representation of the musical sequence, which involves several dimensions. Finally, in [25], Paulus and Klapuri propose a system to transcribe arbitrary drum sounds. The system focuses on the sounds' rhythmic role using clustering techniques and a meter-relative hit description. However, to the best of our knowledge, the presented works have not focused on the expectation of musical audio sig-

nals through the encoding of beat-relative timing and timbre features using a causal approach. Here, we aim to enable the processing of simple percussive material to obtain a system which can be used in the context of modelling musical expectation and memory from both computational and cognitive points of view.

We integrate symbolic expectation models in the system, but we propose in Section 5 some ideas to extend our approach to non-symbolic models. We focus here on percussive musical sequences because (a) methods are available to form a symbolic representation of percussive signals, with a reasonable though far from perfect accuracy [1] (b) percussive sequences can only be learned and expected by solving both *what* and *when* expectation tasks, as opposed to former works focusing on the next pitch prediction [21, 31] without taking into account the timing of the musical events, and (c) modelling learning and expectation of percussive sequences provides a complementary viewpoint with an emphasis on timing and timbre patterns which may help modelling the perception of real-world musical signals along with melodic and tonal approaches.

The system is modelled after the following assumptions: first, it is designed to encode and expect musical hits while analysing signals that exhibit musical periodicity, mainly because the timing is described relative to the extracted beat. That is, our representation requires the attended musical material to exhibit clear beats and onsets. Then, the core processes are supposed to be causal, in order to both respect a strong assumption of how cognitive processes work and allow real-time implementation for interactive music applications.

We present the system components in Section 2. The implementation of the resulting system is presented in Section 3. Preliminary experimental results on a range of audio signals are shown in Section 4. In Section 5, we discuss these preliminary results and propose extensions and alternatives to our approach. Finally, a conclusion is presented in Section 6.

2. SYSTEM

We give here the details of the main components of the system: *listen*, *encode*, *expect*, and *play* modules. These components, all of which run simultaneously when the system is following a musical stream, are shown in Figure 1. First, the *listen* module is the audio front-end, which extracts timbre descriptors, onsets and beats from the incoming signal. This module is based on the aubio library [5]. Each extracted hit is encoded in the *encode* module based on both timbre description and beat-relative inter-onset interval, following an unsupervised scheme. Therefore, we obtain a symbolic representation of the incoming events, to be used by the *expect* module. Based on the generated expectation, the *play* module synthesises a musical event. Several synthesis strategies can be considered depending on the application. We will present some alternatives in Section 2.4.

[1] A comparison of recent techniques to transcribe music audio can be found on `http://www.music-ir.org/mirex2006/`

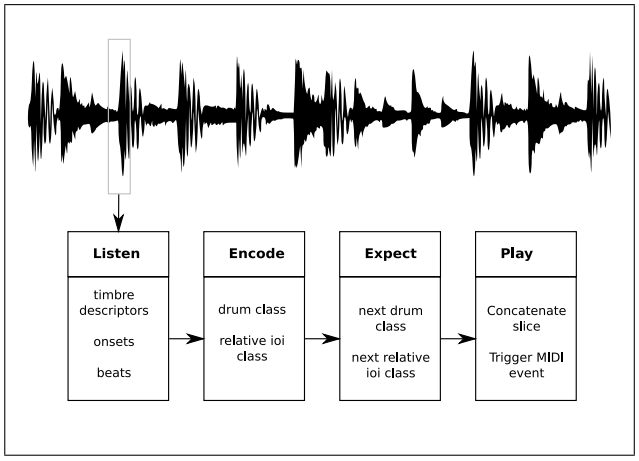

Figure 1. Overview of the system: the incoming audio stream is analysed by the listener module, which provides a transcription (beats, onsets, timbre descriptors) to the encoder module. The encoder produces a reduced representation of this information, which is passed to the expectation module. Based on the produced expectations, the play module generates an output stream.

2.1. Listen module

The listen module uses the *aubio* library [4]. The library provides various functions to extract onset and beat locations, pitch, and other descriptors such as spectral centroid and zero-crossing rate.

2.1.1. Temporal detection

The extraction of onset and beat location is done using the aubio software library [5]. We use a HFC-based detection function and the default parameters for the tempo detection algorithm [9]. We show in Figure 2 the result of the temporal detection process for a musical signal example.

2.1.2. Timbre description

We aim at providing a timbral description of the cues extracted by the onset detector. The timbral descriptors have to be general enough to be applied to a range of musical signals, while they also need to be specific enough to enable the discrimination of perceptually different acoustic cues. As a starting point, we decide to restrict ourselves to a simple description of each hit, based on the following features:

- Zero crossing rate
- Spectral centroid

Note here that the resulting timbre description may be seen as simplistic, as opposed to more refined approaches to timbre description (see [16]). However, this setting has

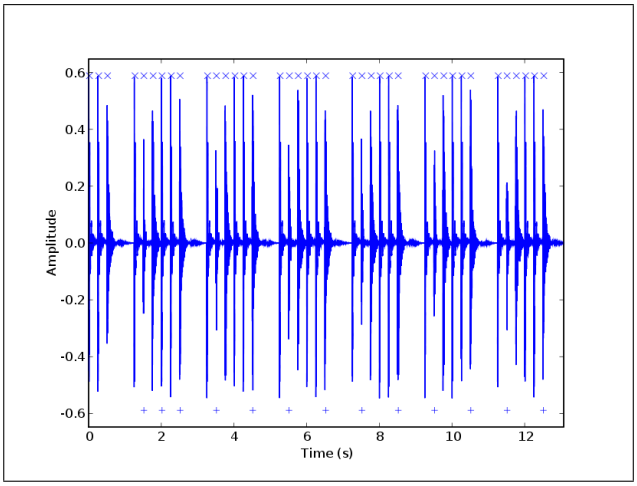

Figure 2. Temporal descriptors obtained on the presented waveform, a simple percussion pattern using three drum instruments. Onsets are marked with x-marks on the top, while beat are marked with crosses at the bottom. The onset detection threshold value is set to 1.

yielded good results when applied to real-time percussive transcription applications [15] involving simple signals (monophonic voice, simple drum instruments). We plan to extend our approach to a wider range of descriptors in future work. Note that the subsequent processes involved in the system work independently of the set of descriptors used.

We will see in Section 4 that the descriptor set we retained here yields limited results when the system deals with more complex signals.

2.2. Encode module

The encode module is responsible for producing a representation from the data (i.e. beats, onset times, and onset descriptors) obtained by the listener module. We aim at producing a symbolic representation of the incoming musical events in order to enable symbolic sequence prediction. However, we plan to extend this approach in future in order to implement sub-symbolic representations (see Section 5). The encode module has two distinct states, *bootstrap* and *running*, which are described as follows.

2.2.1. Bootstrap state

At initialisation time, the system starts from scratch and is exposed to a new musical sequence. Before starting to effectively encode and expect musical events, the system accumulates observations and therefore acts as a short-term memory buffer to accumulate statistics based on the incoming hits. The idea behind this step is that we can take advantage of the time needed (approximately 6s depending of the tempo) by the beat detector to adjust the parameters and distance metrics involved in the running state processes.

Here, we aim to refine the timbre descriptors distance and estimate the number of timbre and inter-onset interval clusters to be represented. Therefore, at the end of the bootstrap state, we normalise the timbre descriptors so that they have a zero mean and a unit variance. Additionally, we rely on the Bayesian Information Criterion (BIC, [30]) to estimate the number of clusters. Note that this latter process, if it helps to obtain a estimate of the number of clusters to encode, is also error prone and non-deterministic. We will have to deal with the resulting variability of clusters estimates when evaluating the system.

2.2.2. Running state

During the running state, the encoder produces a representation of each musical event, based on adjusted parameters and metrics during the bootstrap step. This representation is obtained by quantising and categorising the following event features:

- Timbre descriptors
- Time duration of the previous segment expressed in quarter notes at the current computed tempo, which we name relative inter onset interval.

We use an unsupervised categorisation process to represent the space of input events. We use an online k-means clustering algorithm (as presented in [3]) to split the event space as musical events are presented to the system. When processing a data set, the well known k-means algorithm [14] assigns each instance to a cluster (from a set of fixed size), updates the clusters iteratively, and stops when the cluster assignments converge. The online algorithm we use here does not need to store the instances, but instead updates the clusters in a single pass when processing a new instance. This makes the online k-means *order-dependent*: when dealing with two sequences containing the same items, but in different order, different assignments will be formed. Here we fix the number of clusters at the end of the bootstrap period based on the BIC estimate, we will go back to this point in Section 5.

In Figure 3 (respectively Figure 4), we show the results of encoding the timbre descriptors (respectively the beat-relative interonset intervals) for the example musical signal used in Figure 2.

2.3. Expect module

The expect module has to deduce the most likely future events based on the sequence observed so far. We aim to consider a range of expectation models that we have already applied to symbolic learning tasks [22]. Here, for the sake of compactness, we will restrict our attention to one model. We use a N-gram model, which relies on an exhaustive enumeration of all the possible subsequences and counts their occurrences in the presented data.

N-gram modelling provides a straightforward and efficient approach to probabilistic reasoning and expectation. When a new event x_t is appended in the input sequence, a two-dimensional table, indexing the occurrences of all possible sequences of length n, is incremented by one at

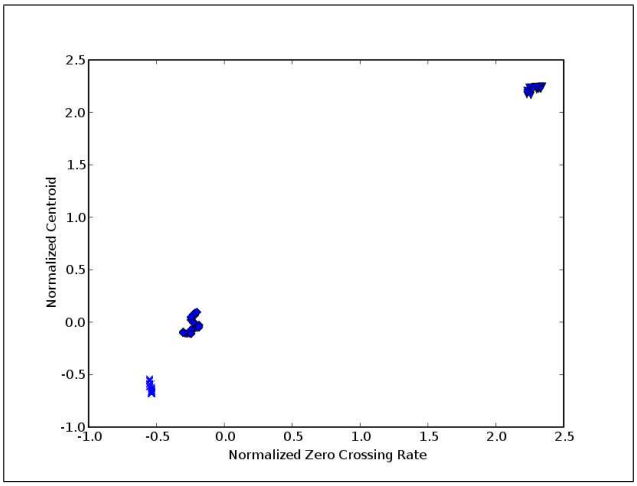

Figure 3. Timbre cluster assignments plotted along the [normalized zcr, normalized centroid] axis. The normalised distance is obtained at the end of the bootstrap step. Here, three clusters are appearing and are marked with triangle, squares and x-marks.

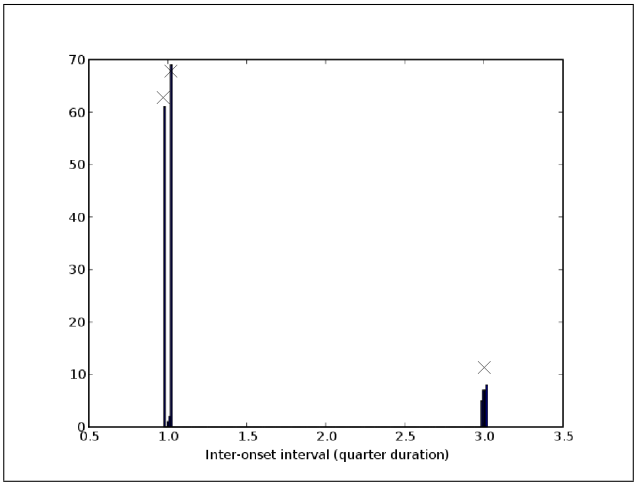

Figure 4. Beat-relative inter-onset interval histogram along with the three cluster centroids, each centroid being annotated with a x-mark. The two clusters around the quarter note correspond to small deviations from an isochronous rhythm.

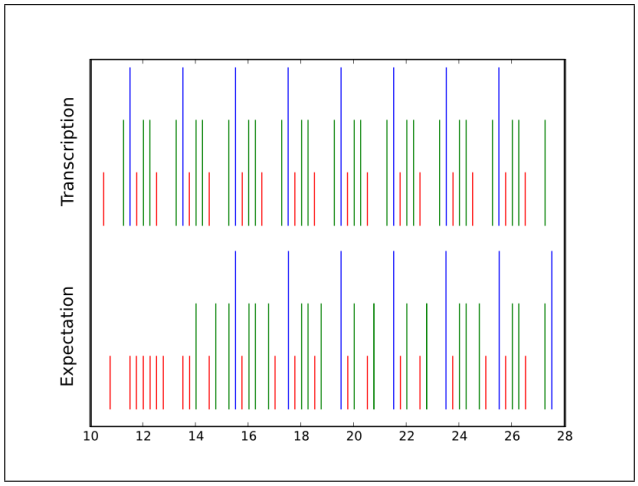

Figure 5. Comparison of encoding (top) and expectation (bottom) of the musical sequence. Ticks from a given cluster have the same height and grey intensity. Each expected tick is produced based on the previous transcribed tick. Note that the system needs a preliminary settling step before providing accurate expectations.

the position with row corresponding to the sequence x_{t-n} ... x_{t-1} and at the column corresponding to the symbol x_t. This means the count table has $nsymbols^N$ cells, where $nsymbols$ is the number of possible symbols. As such, this technique is expensive in terms of space, even if we can use sparse matrix implementations to reduce the size of the transition tables. From the count table and an observation of the actual sequence, we can derive the posterior probability for each possible symbol and produce an expectation accordingly. The N parameter controls the amount of past events considered in the prediction and has to be adjusted carefully. Indeed, an inaccurate choice of N may affect the behaviour of the learner by biasing it to too general predictions (low N), and to over fit prediction (high N). We use here a simple N-Grams model as opposed to more refined approaches in which several models with different context lengths are combined [26]. Overall, this model has the advantage of having fewer parameters than more complex approaches (e.g. recurrent neural networks).

When an expectation is produced, both next hit class and next hit inter-onset interval are computed. We show in Figure 5 the outcome of the expectator module when processing the example used in Figure 2. Based on the generated expectation, we can call the play module.

2.4. Play module

At each time step, the player updates its playing queue: next event playing times are updated by subtracting them the hop size expressed in quarter notes duration at the relative tempo. We propose two strategies for sound generation. First, the system can produce a MIDI event that is used in a software sampler to generate a percussive output stream. To determine which encoded cluster corresponds

to which output sound we sort the drum clusters according to their timbre descriptors means projected along the axis between the more distant cluster means. A second alternative lies in using inter-onset audio slices extracted by the listener module for synthesis, by analogy with [7, 18]. This approach is not cognitively plausible as it implies to store a considerable amount of signal in a buffer. However, this provides an interesting way of evaluating the system qualitatively. When an event from a given cluster has to be played, we look for the slice in which the timbre description is the closest to the mean of that cluster.

3. IMPLEMENTATION

Our current implementation is named *billabio*, as it uses concepts from both aubio [2] and billaboop [3] projects. The components of billabio are written in the Python programming language. For efficiency, we make use of components implemented in C, such as the aubio routines. We make use of an object-oriented approach which closely follows the module description presented in the previous section. The listen module uses aubio to extract timbre descriptors, onsets and beats from the incoming stream. We have modified the aubio core to compute timbre descriptors such as energy, spectral centroid and zero-crossing rate.

Both encode and expect modules, constituting the machine learning algorithms, are implemented using numpy [4]. The online k-means algorithm implemented in the encode module is based on [3], and extended with the bootstrap step we propose in Section 2.2.1. Additionally, we use the *pyem* package [8] to compute the Bayesian Information Criterion and estimate the number of clusters to encode. The online k-means learning rate parameter is set to 0.1, but preliminary experiments show that values in the range [0.1, 0.5] yield similar results in our application domain.

Finally, the N-gram expectator is implemented using pysparse, a sparse matrix module for Python. Two expectation schemes are implemented: deterministic expectation, based on the maximum a posteriori probability, and non-deterministic expectation, based on uniform sampling over the posterior probability of each event. When the N-gram expectator processes an unseen event (i.e. the past sequence of size N have never occurred before), a random prediction based on a uniform distribution over the possible symbols is returned.

4. EXPERIMENTAL RESULTS

In this section we present the method, material, setup we use for characterising the expectation accuracy of the *billabio* framework. Then we present the results of these preliminary experiments.

[2] http://aubio.piem.org
[3] http://www.billaboop.com
[4] http://numpy.scipy.org

4.1. Method

We show here experiments for the *what/when* expectation task. At initialisation, the system starts from scratch and is exposed to an audio excerpt. After the bootstrap step, we can compare the expected hits with the transcribed hits. Note that we do not make use of an annotation of the musical signals, because we first aim to characterise whether the expectator module can provide expectations similar to those that were encoded. Following, we introduce the similarity metric involved in the evaluation.

4.1.1. Similarity metric

Having in mind both encoded and expected signals (see 5), we propose to treat the encoded signal as the ground truth. Consequently, we can compute any statistics usually applied to transcription tasks (e.g. precision, recall and the resulting F-measure) to compare encoding and expectation. As an outcome, we could compute the average F-measure by comparing both transcribed and expected timbre cluster onset times for each of the timbre clusters. This would make the resulting measure very similar to the one used in drum transcription evaluation tasks, e.g. where three drum classes are clearly defined.

However, such metric cannot be considered here because of the inexactness of the unsupervised encoding we use. For instance, the encoder often provides a too large estimate of the number of clusters (e.g. the encoder returns an estimate of five timbre clusters for an excerpt containing three instruments). Consequently, during the running state few of these clusters are indeed used, because a vast majority of the incoming instances are assigned to a subset of the estimated clusters (in the example, three of the five estimated clusters). Consequently, the unused clusters are likely to return very low F-measures, which may in turn affect the average computed F-measure.

To overcome this issue, we define a *weighted average f-measure* as follows:

$$F_w = \sum_{i=1}^{N} w_i F_i \tag{1}$$

where N is the total number of timbre clusters, each w_i is obtained by dividing the number of onsets assigned to cluster i by the total number of onsets, and F_i is the standard F-measure between onsets assigned to cluster i. That is, the individual cluster-wise F-measures involved in the resulting average computation are weighted by the proportion of events appearing in that cluster. This enables to let the unused clusters out of the metric computation. Additionally, each F-measure tolerance window is fixed to 40 ms (20 ms before, 20ms after).

Based on this, a weighted F-Measure evaluates to 1 when all of the processed musical expectations are exactly equal to the transcription (each expected being present within the tolerance window). A weighted F-Measure would equal to zero when none of the onsets expected from a given cluster can be matched with a transcribed onset of

the same cluster. However, we suggest below a more accurate baseline which may enable the comparison of expectation models.

4.2. Finding the baseline: random expectation results

We aim to compare our expectator results with a baseline expectator module that would provide random expectations. That is, given fixed listen and encode modules, we aim at characterising the difference between computing expectations using n-grams and providing random expectations. Indeed, considering this baseline, it is possible to obtain weighted F-measures significantly larger than zero, because the set of symbols representing the musical structure is already computed by the encoder.

4.3. Material

4.3.1. First set: generated material

As a first approach, we consider a set of generated audio signals. We have made the choice of starting evaluating our system with simple material to ensure that the beat detector and onset detector provide an accurate transcription of the signals. Therefore, we have generated a set of 9 audio excerpts using a software drum sequencer. This data is divided into three subsets, generated with two, three, and four instruments. This will enable us to determine whether the encoder is able to retrieve the correct number of timbre clusters. For each of the generated sounds, a unique set of perceptually different instruments is used (e.g. bass drum and hi-hat for a 2 instrument sound). All excerpts contain sixteen beat bars of four beats each.

4.3.2. Second set: acoustic drum recordings

Additionally, we are interested in characterising how the system behaves using a set of acoustic drum recordings. Therefore, we use a selection of 9 drums rhythms from the ENST-Drums database [13]. The excerpts we use are from the following genres: disco, rock, hard rock, shuffle rock, country and funk. Note that the original recordings provided in the database are too short for our system to provide enough expectations after settling. Consequently we produced loops from these excerpts to obtain recordings with an approximate duration of sixteen beat bars of four beats each.

4.4. Setup

For each of the audio signal to be analysed, we run the system using an increasing N-gram length (from 1 to 6), to test whether the results are length-specific. For each of these runs, we run the *billabio* system several times (here 3) because of the stochastic nature of the encoding and expectation processes. We retain the result that is obtained using the optimal N-gram length. Consequently, each musical excerpt presented in last section is analysed 18 times. Additionally, to test the random expectator condition, we run the system 3 times for each file. This leads to a total number of runs equal to 189 for each set, totalising approximately 100 minutes of audio to be analysed. The average runtime of each experiment is 50 minutes on a Pentium 4 Ubuntu laptop with 512 Mb of memory.

Excerpt	n-gram	random
4 timbres 1	0.68	0.19
2 timbres 3	0.66	0.15
2 timbres 1	0.65	0.49
3 timbres 2	0.65	0.42
3 timbres 1	0.62	0.18
3 timbres 3	0.55	0.19
4 timbres 3	0.44	0.22
4 timbres 2	0.36	0.15
2 timbres 2	0.34	0.26

Table 1. Expectation experiment results using the generated rhythm set. The post-exposure weighted F-measure is shown under the following conditions: n-gram based expectator and uniform random guess.

4.5. Results

In Table 1 and Table 2, we present the results of the experiment presented above for the first and second set. Concerning the first set (generated material), we observe an average weighted F-measure of 0.55, ranging from 0.34 to 0.66. The random condition baseline gives an average weighted F-measure of 0.25, ranging from 0.15 to 0.49. Concerning the second set (acoustic drums recordings), we observe an average weighted F-measure of 0.59, ranging from 0.25 to 0.96. The random condition baseline gives an average weighted F-measure of 0.32, ranging from 0.18 to 0.56. Surprisingly, the overall results for the second set are slightly better than those of the first set, however they also exhibit more variability (the system reaches the score of 0.96 for the disco 2 excerpt while only attaining a score of 0.25 for the funk 2 excerpt). Overall, the n-gram expectator outperforms the random expectator by a factor of approximately 2.

As a side note, it is important to note that among the runs described, only 51 runs out of 189 led to an accurate estimate of the number of clusters. This shows why our task requires to effectively integrate uncertainty in our model and performance metrics.

5. DISCUSSION AND FUTURE WORK

The model is already able to learn aspects of the musical audio, however we can see that there is room for improvement. It is difficult to compare our results with existing approaches because few works have focused on the expectation in both timing and timbre of the attended musical stream. Overall, we consider the figures presented here as a baseline for future works in this area. Our simple N-gram expectation algorithm leads to a weighted F-measure outperforming the random expectator by a factor

Excerpt	n-gram	random
disco 2	0.96	0.56
disco 1	0.90	0.44
shuffle	0.79	0.19
rock 1	0.78	0.47
country	0.60	0.36
hard	0.46	0.31
funk 1	0.34	0.22
rock 2	0.31	0.18
funk 2	0.25	0.21

Table 2. Expectation experiment results using the acoustic drums recordings set. The post-exposure weighted F-measure is shown under the following conditions: n-gram based expectator and uniform random guess.

of approximately two. However, some excerpt are poorly represented and expected. Further, we are not able to show a clear relation between N-gram length, number of clusters and expectation results. Overall, we are now able to compare different expectation algorithms, and we plan to incorporate in our comparison other types of algorithms such as recurrent neural networks and multiresolution N-grams (see Section 1).

On the other hand, the timbre description we have presented here may be seen as simplistic. We will consider incrementing the number of timbre features. We will use data transformation techniques such as Principal Component Analysis instead of simple parameter normalisation to deal successfully with an increased number of descriptors. The next descriptor we plan to include is pitch, that is, the fundamental frequency extracted over an inter onset interval region. This will enable to apply our system to a wider range of musical signals.

Then, given that real world musical sequences do not have uniform distributions of instrument and inter-onset intervals along the whole sequence, it may be preferable to complete a bootstrap computation in parallel with the other processes that take place during the running state. That is, the cluster estimate could be adjusted during the running state, and would for instance enable to get rid of unused clusters. We have pointed out that we transcribe here the audio stream into a symbolic sequence of events, each of which has a *hard* assignment to a given cluster. We aim to operate a move towards sub-symbolic processing by encoding the incoming events with *soft* cluster assignments, in which each event has *membership* values associated with each cluster ([20]). We will need to adapt the expect module to this new representation. Finally, we will ultimately need to compare the system expectation with ground truth data instead of the transcription to assess how well the system matches human perception and expectation.

6. CONCLUSION

We have presented a framework to transcribe, encode, generate, and synthesise expectations based on constant-tempo musical audio. First, a review of the existing approaches related to our work, in the fields of audio transcription, sequence learning and expectation, and interactive music systems has been presented. We have then proposed a system architecture and detailed each of its components, presenting our main assumptions, and proposing alternatives to music expectation and generation. Our current implementation, *billabio* has been described. We have then proposed an evaluation metric derived from earlier works on drum transcription that takes into account the stochastic nature of the unsupervised encoding we use. Experimental results obtained using different audio recordings have been shown and a baseline based on random expectation has been proposed and computed. Finally, we have discussed our approach and the preliminary results we obtained. Further, we have proposed extensions to overcome some assumptions presented here. We aim at refining our approach in order to obtain a system enabling both interaction and real-world application for music representation and expectation models. Sound examples are available on `http://www.iua.upf.edu/~ahazan/billabio`.

7. ACKNOWLEDGEMENTS

This work has been partially funded by the *EmCAP* project (European Commission FP6-IST contract 013123).

8. REFERENCES

[1] C. Allauzen, M. Crochemore, and M. Raffinot. Factor oracle: A new structure for pattern matching. In *Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics on Theory and Practice of Informatics*, pages 295 – 310, 1999.

[2] G. Assayag and S. Dubnov. Using Factor Oracles for machine Improvisation. *Soft Computing*, 8(9):604–610, 2004.

[3] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[4] P. Brossier. *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Centre for Digital Music, Queen Mary University of London, London, UK, Sept. 2006.

[5] P. Brossier, J. Bello, and M. Plumbley. Fast labelling of notes in music signals. In *Proceedings of the 5th International Symposium on Music Information Retrieval (ISMIR 2004)*, pages 331–336, Barcelona, Spain, 2004.

[6] A. T. Cemgil. *Bayesian Music Transcription*. PhD thesis, Radboud University of Nijmegen, The Netherlands, 2004.

[7] N. Collins. On onsets on-the-fly: Real-time events segmentation and categorization as a compositional effect. In *Proceedings of the First Sound and Music Computing Conference (SMC'04)*, Paris, France, 2004.

[8] D. Cournapeau. Pyem, a python package for gaussian mixture models. Technical report, University of Kyoto, Graduate School of Informatics, 2006.

[9] M. E. P. Davies, P. M. Brossier, and M. D. Plumbley. Beat tracking towards automatic musical accompaniment. In *Proceedings of the Audio Engineering Society 118th convention*, Barcelona, Spain, May 2005.

[10] D. Eck and J. Schmidhuber. Learning the long-term structure of the blues. *Lecture Notes in Computer Science, Proceedings of ICANN Conference*, 2415:284–289, 2002.

[11] J. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, Apr. 1990.

[12] M. Ferrand, P. Nelson, and G. Wiggins. A probabilistic model for melody segmentation. In *2nd International Conference on Music and Articial Intelligence (IC-MAI2002)*, University of Edinburgh, UK, 2002.

[13] O. Gillet and G. Richard. Enst-drums: an extensive audio-visual database for drum signals processing. In *Proceedings of the 7th International Symposium on Music Information Retrieval (ISMIR 2006)*, pages 156–159, October 2006.

[14] J. Hartigan and M. Wong. Algorithm as 136: A k-means clustering algorithm. *Applied Statistics*, 28, 1979.

[15] A. Hazan. Billaboop: Real-time voice-driven drum generator. In *Proceedings of Audio Engineering Society, 118th Convention*, 2005.

[16] P. Herrera, A. Dehamel, and F. Gouyon. Automatic labeling of unpitched percussion sounds. In *Proceedings of the 114th Audio Engineering Society Convention*, Amsterdam, The Netherlands, 2003.

[17] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] T. Jehan. *Creating Music by Listening*. PhD thesis, Massachusetts Institute of Technology, MA, USA, Sept. 2005.

[19] O. Lartillot, S. Dubnov, G. Assayag, and G. Bejerano. Automatic modeling of musical style. In *Proceedings of the International Computer Music Conference (ICMC 2001)*, La Havana, Cuba, 2001.

[20] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Publisher: Cambridge University Press, 2003. Available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

[21] L. C. Manzara and D. Conklin. Comparing human and computational, 1995.

[22] R. Marxer, A. Hazan, H. Purwins, M. Grachten, P. Herrera, and I. Salselas. Mock-up of music analysis system. Technical report, Music Technology Group, Pompeu Fabra University, 2007.

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

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

[25] J. Paulus and A. Klapuri. Model-based event labeling in the transcription of percussive audio signals. In M. Davies, editor, *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, pages 73–77, London, UK, Sept. 2003.

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

[27] C. Raphael. Music plus one: A system for flexible and expressive musical accompaniment. In *Proceedings of the International Computer Music Conference*, La Havana, Cuba, 2001.

[28] D. Rumelhart and J. McLelland. *Parallel Distributed Processing*. MIT Press, 1986.

[29] D. Schwarz. *Data-Driven Concatenative Sound Synthesis*. PhD thesis, Ircam - Centre Pompidou, Paris, France, Jan. 2004.

[30] G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, Mar. 1978.

[31] I. H. Witten, L. C. Manzara, and D. Conklin. Comparing human and computational models of music prediction. Technical Report 92/477/15, University of Calgary, 1992.