Labelling the Structural Parts of a Music Piece with Markov Models

Jouni Paulus and Anssi Klapuri

Department of Signal Processing, Tampere University of Technology
Korkeakoulunkatu 1, Tampere, Finland
{jouni.paulus, anssi.klapuri}@tut.fi

Abstract. This paper describes a method for labelling structural parts of a musical piece. Existing methods for the analysis of piece structure often name the parts with musically meaningless tags, e.g., "p1", "p2", "p3". Given a sequence of these tags as an input, the proposed system assigns musically more meaningful labels to these; e.g., given the input "p1, p2, p3, p2, p3" the system might produce "intro, verse, chorus, verse, chorus". The label assignment is chosen by scoring the resulting label sequences with Markov models. Both traditional and variable-order Markov models are evaluated for the sequence modelling. Search over the label permutations is done with N-best variant of token passing algorithm. The proposed method is evaluated with leave-one-out cross-validations on two large manually annotated data sets of popular music. The results show that Markov models perform well in the desired task.

1 Introduction

Western popular music pieces often follow a sectional form in which the piece is constructed from shorter units. These units, or musical parts, may have distinct roles on the structure of the piece, and they can be named based on this role, for example, as "chorus" or "verse". Some of the parts may have several occurrences during the piece (e.g., "chorus") while some may occur only once (e.g., "intro").

To date, several methods have been proposed to perform automatic analysis of the structure of a musical piece from audio input, see [1] or [2] for a review. Majority of the methods do not assign musically meaningful labels to the structural parts they locate. Instead, they just provide information about the order, possible repetitions, and temporal boundaries of the found parts. There also exist a few methods that utilise musical models in the analysis, and the resulting structure descriptions have musically meaningful labels attached to the found parts [3,4].

The musical piece structure can be used, for example, in a music player user interface allowing the user to navigate within the piece based on musical parts [5]. The results of a user study with a music player having such a navigation ability suggest that the parts should be labelled meaningfully. The additional information of knowing which of the parts is for instance "chorus" and which is "solo" was judged to be valuable [6].

The proposed method does not perform the musical structure analysis from audio, but only labels structural descriptions and should be considered as an add-on or an extension to existing structure analysis systems. So, the problem to be solved here is

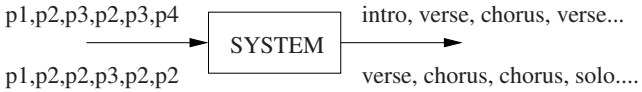

Fig. 1. Basic idea of the system. The system assigns meaningful labels to arbitrary tags based on a musical model. The mapping from tags to labels is determined separately for each input.

how to assign musically more meaningful part *labels* when given a sequence of *tags* describing the structure of a musical piece. The operation is illustrated in Figure 1. As an example, the structure of the piece "Help!" by The Beatles is "intro, verse, refrain, verse, refrain, verse, refrain, outro", as given in [7]. A typical structure analysis system might produce "p1,p2,p3,p2,p3,p2,p3,p4" as the result, which then would be the input to the proposed system. If the system operation was successful, the output would be the assignment: "p1 → intro, p2 → verse, p3 → refrain, p4 → outro".

It is often said more or less seriously that popular music pieces tend to be of the same form, such as "intro, verse, chorus, verse, chorus, solo, chorus".[1] The proposed method aims to utilise this stereotypical property by modelling the sequential dependencies between part labels (occurrences of musical parts) with Markov chains, and searching the label assignment that maximises the probability of the resulting label sequence. Evaluation show that the sequential dependencies of musical parts are so informative that they can be used in the labelling.

The rest of the paper is structured as following: Sect. 2 describes the proposed method. The labelling performance of the method is evaluated in Sect. 3. Sect. 4 gives the conclusions of the paper.

2 Proposed Method

The input to the system is a sequence of tags $R_{1:K} \equiv R_1, R_2, \ldots, R_K$, and the problem is to assign a musical label to each of the unique tags so that no two tags are assigned the same label. This assignment is defined as an injective mapping function $f : T \rightarrow L$ from input set T of tags to the output set L of musically meaningful labels, as illustrated in Figure 2. The mapping function transforms the input tag sequence $R_{1:K}$ into a sequence of labels $f(R_{1:K}) = S_{1:K}$.

The proposed methods assumes that the musical parts depend sequentially on each other in the form of a Markov chain and that it is possible to predict the next musical part given a finite history of the preceding parts. This predictability is used to score different mapping alternatives and the best mapping is then given as the output of the system.

[1] Though statistics from two data sets of popular music pieces show that the structures of the pieces are more heterogeneous than was initially expected, see Sect. 3.1.

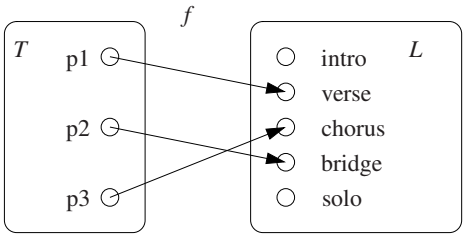

Fig. 2. An example of the mapping function $f : T \rightarrow L$. All tags in T are mapped to one label in L, but some labels in L may remain unused.

2.1 Markov Models

Markov models assume that the probability of a continuation S_i for sequence $S_{1:(i-1)}$ depends only on a finite history of the sequence $S_{(i-(N-1)):(i-1)}$ instead of the full history, i.e., $p(S_i|S_{1:(i-1)}) = p(S_i|S_{(i-(N-1)):(i-1)})$, where $(N-1)$ is the length of the used history. This is also referred as the order of the resulting Markov model and gives rise to the alternative name of N-grams. Based on the Markov assumption, the overall probability of a sequence $S_{1:K}$ is obtained by

$$p(S_{1:K}) = \prod_{k=1}^{K} p(S_k|S_{(k-(N-1)):(k-1)}) \ . \tag{1}$$

In the beginning of the sequence where there is not enough history available, it is possible to use a lower order model or pad the sequence from the beginning with a special symbol. [8]

The total N-gram probability of (1) is used to score different mapping functions by evaluating it for the output sequences after the mapping $f(R_{1:K}) = S_{1:K}$. The target is to find the mapping function f_{OPT} that maximises the total probability

$$f_{OPT} = \underset{\text{f}}{\text{argmax}}\{p(f(R_{1:K}))\},\ f : T \rightarrow L \text{ injective} \ . \tag{2}$$

2.2 Optimisation Algorithm

The maximisation problem is solved by using M-best[2] variant of token passing (TP) algorithm, more frequently used in speech recognition [9]. The main principle of TP is that tokens t are propagated time synchronously between the states of the model. Each token knows the path it has travelled and accumulates the total probability over it. Based on the path probabilities, the M tokens with the highest probabilities are selected for propagation in each state, they are replicated and passed to all connected states. The token path probabilities are updated based on the transition probabilities between the states.

[2] Better known as the N-best token passing. The name is adjusted to avoid possible confusion with N-grams.

The state space of TP is formed from the possible labels in L, and the paths of the tokens encode different mapping functions. The optimisation of (2) can be done by searching the most probable path through the states (labels) defining the state transition probabilities with

$$p\left(f_k(R_k) = l_i | R_{1:(k-1)}, f_{k-1}\right) = \begin{cases} 0, & \text{if } \texttt{DME} \\ p(l_i | f_{k-1}(R_{1:(k-1)})), & \text{otherwise} \end{cases}, \tag{3}$$

where DME denotes the predicate "different mapping exists", which is used to guarantee that the mapping function is injective, and it is defined by

$$\texttt{DME} = \exists j : (R_j = R_k \wedge f_{k-1}(R_j) \neq l_i) \vee (R_j \neq R_k \wedge f_{k-1}(R_j) = l_i), j \in [1, k-1] \ . \tag{4}$$

In the equations above, $p\left(f_k(R_k) = l_i | R_{1:k}, f_{k-1}\right)$ denotes the probability of a token to transition to the state corresponding to label l_i after it has travelled the path $f_{k-1}(R_{1:(k-1)})$. The N-gram probability for label l_i when the preceding context is $f_{k-1}(R_{1:(k-1)})$, is denoted as $p(l_i | f_{k-1}(R_{1:(k-1)}))$. As the mapping is generated gradually, f_k is used to denote the mapping after handling the sequence $R_{1:k}$.

Pseudocode of the algorithm is given in Algorithm 1. It searches a mapping function $f : T \rightarrow L$ from tags in input sequence to the possible label set. For each label $l \in L$, the probability $\pi_0(l)$ of that label to be the first label in the sequence and the probability the label the be the last $\pi_E(l)$ are defined. In the middle of the sequence, the probability of the continuation given the preceding context is obtained from (3).

As the mapping depends on decisions done within the whole preceding history, the Markov assumption is violated and the search cannot be done with more efficient methods guaranteeing a globally optimal solution. This sub-optimality hinders also the traditional TP, since it might be that the optimal labelling may not be the best one earlier in the sequence, and is therefore pruned during the search. The M-best variant of TP alleviates this problem by propagating M best tokens instead of only the best one. If all tokens were propagated, the method would find the globally optimal solution, but at a high computational cost. With a suitable number of tokens, a good result can be found with considerably less computation than with an exhaustive search. An exhaustive search was tested, but due to the large search space, it proved to be very inefficient. However, it was used to verify the operations of TP with a subset of the data. In that subset, the TP showed to find the same result as the exhaustive search in almost all the cases when storing 100 tokens at each state.

2.3 Modelling Issues

The main problem with N-gram models is the amount of training material needed for estimating the transition probabilities: the amount increases rapidly as a function of the number of possible states and the length of used history (given A possible states and history length of N, there exist A^N probabilities to be estimated). It may happen that not all of the sequences of the required length occur in the training data. This situation can be handled by back-off (using shorter context at those cases), or by smoothing (assigning a small amount of the total probability mass to the events not encountered in the training material).

Algorithm 1: Search label mapping $f : T \to L$

Input sequence $R_{1:K}$.
Label space L. Associated with each label $l \in L$, there are input buffer I_l and output buffer O_l.
Tokens t with probability value $t.p$ and label mapping function $t.f$.
for $l \in L$ **do** // initialisation
  Insert t to I_l and assign $t.p \leftarrow \pi_0(l)$
for $k \leftarrow 1$ **to** K **do**
  for $l \in L$ **do**
    $O_l \leftarrow I_l$ // propagate to output
    Clear I_l
  for $l \in L$ **do** // transition source
    for $t \in O_l$ **do**
      for $\tilde{l} \in L$ **do** // transition target
        $\tilde{t} \leftarrow t$ // copy token
        if $\exists j : R_j = R_k, j \in [1, k-1]$ **then**
          if $\tilde{t}.f(R_k) = \tilde{l}$ **then**
            $\tilde{t}.p \leftarrow \tilde{t}.p \times p(\tilde{t}.f(R_k)|\tilde{t}.f(R_{1:(k-1)}))$ // N-gram probability
          else
            $\tilde{t}.p \leftarrow 0$
        else
          if $\forall j : \tilde{t}.f(R_j) \neq \tilde{l}, j \in [1, k-1]$ **then**
            Set $\tilde{t}.f(R_k) \leftarrow \tilde{l}$
            $\tilde{t}.p \leftarrow \tilde{t}.p \times p(\tilde{t}.f(R_k)|\tilde{t}.f(R_{1:(k-1)}))$
          else
            $\tilde{t}.p \leftarrow 0$
        Insert $\tilde{t}$ to $I_{\tilde{l}}$
  for $l \in L$ **do**
    Retain M best tokens in I_l
for $l \in L$ **do**
  $O_l \leftarrow I_l$
  for $t \in O_l$ **do**
    $t.p \leftarrow t.p \times \pi_E(l)$
Select token $\hat{t}$ with the largest $t.p$
return $\hat{t}.f$

In some cases, it is possible that increasing the length of the context does not provide any information compared to the shorter history. Variable-order Markov models (VMMs) have been proposed to replace traditional N-grams. Instead of using a fixed history, VMMs try to deduce the length of the usable context from the data. If increasing the length of the context does not improve the prediction, then only the shorter context is used. VMMs can be used to calculate the total probability of the sequence in the same manner as in (1), but using a variable context length instead of fixed N. [10]

3 Evaluations

Performance of the labelling method was evaluated in simulations using structural descriptions from real music pieces.

3.1 Data

The method was evaluated on two separate data sets. The first, *TUTstructure07*, was collected at Tampere University of Technology. The database contains a total of 557 pieces sampling the popular music genre from 1980's to present day.[3] The musical structure of each piece was manually annotated. The annotation consists of temporal segmentation of the piece into musical parts and naming each of the parts with musically meaningful labels. The annotations were done by two research assistants with some musical background. The second data set, *UPF Beatles*, consists of 174 songs by The Beatles. The original piece structures were annotated by musicologist Alan W. Pollack [7], and the segmentation time stamps were added at Universitat Pompeu Fabra (UPF)[4].

Though many of the forms are thought to be often occurring or stereotypical for music from pop/rock genre, the statistics from the data sets do not support this fully. In TUTstructure07, the label sequences vary a lot. The three most frequently occurring structures are

- "intro", "verse", "chorus", "verse", "chorus", "C", "chorus", "outro"
- "intro", "A", "A", "B", "A", "solo", "B", "A", "outro"
- "intro", "verse", "chorus", "verse", "chorus", "chorus", "outro",

each occurring four times in the data set. 524 (94%) of the label sequences are unique.

With UPF Beatles, there is a clearer top, but still there is a large body of sequences occurring only once in the data set. The most frequent label sequence is

- "intro", "verse", "verse", "bridge", "verse", "bridge", "verse", "outro",

occurring seventeen times in the data set. 135 (78%) of the label sequences are unique.

3.2 Training the Models

Transition probabilities for the models were trained using the data sets. Each label sequence representing the structure of a piece was augmented with special labels "BEG" in the beginning, and "END" in the end. After the augmentation, the total number of occurrences of each label in the data set was counted. Because there exists a large number of unique labels, some of which occur only once in the whole data set, the size of the label alphabet was reduced by using only the labels that cover 90% of all occurrences. The remaining labels were replaced with an artificial label "MISC". The zero-probability problem was addressed by using Witten-Bell discounting (Method C in [11]), except for the VMMs.

[3] List of the pieces is available at <http://www.cs.tut.fi/sgn/arg/paulus/TUTstructure07_files.html>.

[4] <http://www.iua.upf.edu/%7Eperfe/annotations/sections/license.html>

In the original data sets, there were 82 and 52 unique labels (without the augmentation labels "BEG", "END", and "MISC") in the data set of TUTstructure07 and UPF Beatles, respectively. After augmentation and set reduction the label set sizes were 15 and 10. On the average, there were 6.0 unique labels and 12.1 label occurrences (musical parts) in a piece in TUTstructure07. The same statistics for UPF Beatles were 4.6 and 8.6. This suggests that the pieces in TUTstructure07 were more complex or they have been annotated on a finer level.

3.3 Simulation Setup

In simulations, the structural annotations from the data base were taken. The original label sequences (with the "MISC" substitution) was taken as the ground truth, while the input to the labelling algorithm was generated by replacing the labels with letters.

To avoid overlap in train and test sets whilst utilising as much of the data as possible, simulations were run using leave-one-out cross-validation scheme. In each cross-validation iteration one of the pieces in the data set was left as the test case while the Markov models were trained using all the other pieces. This way the model never saw the piece it was trying to label.

With conventional N-grams, the length of the Markov chain was varied from 1 to 5, i.e., from using just prior probabilities for the labels to utilising context of length 4. With VMMs, several different algorithms were tested, including: decomposed context tree weighting (DCTW), prediction by partial matching - method C, and a variant of Lempel-Ziv prediction algorithm. The implementations for these were provided by [12]. It was noted that DCTW worked the best of these three, and the result are presented only for it. The maximum context length for VMMs was set to 5. Also the maximum context lengths of 3 and 10 were tested, but the former deteriorated the results and the latter produced practically identical results with the chosen parameter value.

3.4 Evaluation Metrics

When evaluating the labelling result, confusion matrix C for the labels is calculated. The result of the best mapping function applied to the input sequence $f(R_{1:K})$ and the ground truth sequence $S_{1:K}$ are compared. At each label occurrence $S_i, i \in [1,K]$, the value in the element $[S_i, f(R_i)]$ of the confusion matrix is increased by one. This applies weighting for the more frequently occurring labels. The confusion matrix is calculated over all cross-validation iterations. The average hit rate for a target label was calculated as a ratio of correct assignments (main diagonal of confusion matrix) to total occurrences of the label (sum along rows of the confusion matrix).

3.5 Results

The effect of varying the context length in N-grams is shown in Tables 1 and 2 for TUTstructure07 and UPF Beatles, respectively. In addition to the different N-gram lengths, the tables contain also the result for the best VMM (DCTW with maximum memory length of 5). The tables contain the percentage of correct assignments for each label

Table 1. Performance comparison on TUTstructure07 with traditional Markov models of different order. The best VMM result is given for comparison. The given values are the average hit rates in percents. The row *average* is the total average of correct part labels. The best result on each row is typeset with bold.

label	N=1	N=2	N=3	N=4	N=5	VMM
chorus	68.1	76.3	**80.8**	76.6	74.9	78.5
verse	42.3	62.4	64.4	64.9	**66.0**	66.0
bridge	17.7	38.6	45.6	**47.4**	44.4	43.7
intro	27.6	97.6	**98.2**	97.8	97.8	96.4
pre-verse	4.2	40.7	**46.3**	43.3	41.7	43.3
outro	13.9	98.3	**98.6**	97.8	92.1	98.3
c	0.0	38.0	42.1	47.4	**54.8**	49.3
theme	0.0	0.0	2.7	**4.4**	3.3	3.3
solo	0.0	4.4	7.2	16.0	**18.2**	14.9
chorus_a	0.0	0.0	7.5	**15.7**	11.2	3.0
a	0.0	0.0	**32.5**	31.7	27.0	29.4
chorus_b	0.0	0.9	5.3	**12.4**	7.1	2.7
MISC	12.6	29.5	38.3	37.1	**40.3**	38.3
average	30.9	55.6	**60.3**	59.9	59.5	59.8

used. The total average of correct hits ("average") is calculated without the augmentation labels "BEG" and "END".[5]

Based on the results in Tables 1 and 2, it can be seen that increasing the order of traditional Markov model from unigrams to bigrams produce a large increase in the performance. The performance continues to increase when the context length is increased, but more slowly. With TUTstructure07, the performance peak is at $N = 3$, whereas with UPF Beatles, the maximum with traditional N-grams can be obtained with $N = 4$. It was also noted that with TUTstructure07 the use of VMM did not improve the result. However, there is a small performance increase with VMMs in UPF Beatles.

Even though the use of VMM did not improve the result with TUTstructure07, there was one clear advantage with them: it was possible to use longer context in the models. With traditional N-grams, the transition probabilities will become very sparse even with bigrams. The large blocks of zero provide no information whatsoever and only consume memory. With VMMs, the context length is adjusted according to the available information.

From the results, it is notable that "chorus" can be labelled from the input over 80% accuracy, and "verse" almost at 65% accuracy in TUTstructure07. In UPF Beatles "verse" could be labelled with 87% accuracy and "refrain" with 71% accuracy.

[5] For an interested reader, the confusion matrices are given in a document available at <http://www.cs.tut.fi/sgn/arg/paulus/CMMR08_confMats.pdf>.

Table 2. Performance comparison on UPF Beatles with traditional Markov models of different order. The best VMM result is given for comparison. For description of the data, see the Table 1.

label	N=1	N=2	N=3	N=4	N=5	VMM
verse	72.4	79.9	86.7	85.7	83.7	**87.5**
refrain	30.1	32.1	62.2	66.3	68.7	**70.7**
bridge	36.7	40.7	**78.0**	74.0	74.0	70.6
intro	0.0	93.2	88.9	92.0	**93.8**	93.2
outro	0.0	99.3	**99.3**	97.2	93.0	97.9
verses	0.0	16.1	48.2	**50.0**	44.6	44.6
versea	0.0	5.9	7.8	17.6	**21.6**	5.9
MISC	0.0	15.9	22.3	**25.5**	23.6	22.3
average	33.5	58.9	72.1	72.8	72.1	**73.0**

3.6 Discussion

It should be noted that the proposed system performs the labelling purely based on a model of sequential dependencies of musical parts. Incorporating some acoustic information might improve the result somewhat (e.g., energetic repeated part might be "chorus"). Also, the knowledge of the high-level musical content, such as the lyrics, instrumentation or chord progressions, could provide valuable information for the labelling. However, the extraction of these from the acoustic input is still a challenging task, as well as creating a usable model for them. In addition, when discussing the principles used when assigning the ground truth labels with the annotators, the main cue was the location of the part in the "musical language model". Incorporating these other information sources in addition to the sequence model should be considered in the future work.

The difference in performance between the two data sets remains partly an open question. The main reason may be that the label sequences in TUTstructure07 are more diverse, as could be seen from the statistics presented in Sec. 3.1 (94% of sequences in TUTstructure are unique, compared to 78% in UPF Beatles). We tested the hypothesis that is was due to the smaller label set (10 vs. 15) by using only as many of the most frequent labels as were used with UPF Beatles. As a slight surprise, the performance on the remaining set was even worse compared label-wise to the larger set. The average result, however, increased slightly because the most rarely occurring (and most often mis-labelled) labels were omitted.

4 Conclusion

This paper has proposed a method for assigning musically meaningful labels to the parts found by music structure analysis systems. The method models the sequential dependencies between musical parts with Markov models and uses the models to score different label assignments. The paper has proposed applying M-best token passing algorithm to the label assignment search to be able to perform the assignment without

having to test all possible permutations. The proposed method has been evaluated with leave-one-out cross-validations on two data sets of popular music pieces. The evaluation results suggest that the models for the sequential dependencies of musical parts are so informative even at low context lengths that they can be used alone for labelling. The obtained labelling performance was reasonable, even though the used model was relatively simple.

Acknowledgements

This work was supported by the Academy of Finland, project No. 5213462 (Finnish Centre of Excellence program 2006 - 2011). The TUTstructure07 was annotated by Toni Mäkinen and Mikko Roininen.

References

1. Peeters, G.: Deriving musical structure from signal analysis for music audio summary generation: "sequence" and "state" approach. In: Lecture Notes in Computer Science. Volume 2771. Springer-Verlag (2004) 143–166
2. Ong, B.S.: Structural analysis and segmentation of musical signals. PhD thesis, Universitat Pompeu Fabra, Barcelona (2006)
3. Shiu, Y., Jeong, H., Kuo, C.C.J.: Musical structure analysis using similarity matrix and dynamic programming. In: Proc. of SPIE Vol. 6015 - Multimedia Systems and Applications VIII. (2005)
4. Maddage, N.C.: Automatic structure detection for popular music. IEEE Multimedia **13**(1) (January 2006) 65–77
5. Goto, M.: A chorus-section detecting method for musical audio signals. In: Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Hong Kong (2003) 437–440
6. Boutard, G., Goldszmidt, S., Peeters, G.: Browsing inside a music track, the experimentation case study. In: Proc. of 1st Workshop on Learning the Semantics of Audio Signals, Athens (December 2006) 87–94
7. Pollack, A.W.: 'Notes on...' series. The Official rec.music.beatles Home Page (http://www.recmusicbeatles.com) (1989–2001)
8. Jurafsky, D., Martin, J.H.: Speech and language processing. Prentice-Hall, New Jersey, USA (2000)
9. Young, S.J., Russell, N.H., Thornton, J.H.S.: Token passing: a simple conceptual model for connected speech recognition systems. Technical Report CUED/F-INFENG/TR38, Cambridge University Engineering Department, Cambridge, UK (July 1989)
10. Ron, D., Singer, Y., Tishby, N.: The power of amnesia: Learning probabilistic automata with variable memory length. Machine Learning **25**(2–3) (1996) 117–149
11. Witten, I.H., Bell, T.C.: The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transcations on Information Theory **37**(4) (1991) 1085–1094
12. Begleiter, R., El-Yaniv, R., Yona, G.: On prediction using variable order Markov models. Journal of Artificial Intelligence Research **22** (2004) 385–421