deepGTTM-I&II: Local Boundary and Metrical Structure Analyzer based on Deep Learning Technique

Masatoshi Hamanaka[1], Keiji Hirata[2], and Satoshi Tojo[3]

[1] Kyoto University
masatosh@kuhp.kyoto-u.ac.jp
[2] Future University Hakodate
hirata@fun.ac.jp
[3] JAIST
tojo@jaist.ac.jp

Abstract. This paper describes an analyzer for detecting local grouping boundaries and generating metrical structures of music pieces based on a generative theory of tonal music (GTTM). Although systems for automatically detecting local grouping boundaries and generating metrical structures, such as the full automatic time-span tree analyzer, have been proposed, musicologists have to correct the boundaries or strong beat positions due to numerous errors. In light of this, we use a deep learning technique for detecting local boundaries and generating metrical structures of music pieces based on a GTTM. Because we only have 300 pieces of music with the local grouping boundaries and metrical structures analyzed by musicologist, directly learning the relationship between the scores and metrical structures is difficult due to the lack of training data. To solve this problem, we propose a multi-task learning analyzer called deepGTM-I&II based on the above deep learning technique to learn the relationship between scores and metrical structures in the following three steps. First, we conduct unsupervised pre-training of a network using 15,000 pieces of music in a non-labeled dataset. After pre-training, the network involves supervised fine-tuning by back propagation from output to input layers using a half-labeled dataset, which consists of 15,000 pieces of music labeled with an automatic analyzer that we previously constructed. Finally, the network involves supervised fine-tuning using a labeled dataset. The experimental results indicate that deepGTTM-I&II outperformed previous analyzers for a GTTM in terms of the F-measure for generating metrical structures.

Keywords: generative theory of tonal music (GTTM), local grouping boundary, grouping structure, metrical structure, deep learning.

1 Introduction

We propose an analyzer for automatically detecting local grouping boundaries and generating metrical structures of music pieces based on a generative theory

of tonal music (GTTM) [1]. A GTTM is composed of four modules, each of which assigns a separate structural description to a listener's understanding of a piece of music. These four modules output a grouping structure, metrical structure, time-span tree, and prolongational tree. Since the detection of the local grouping boundaries and generation of metrical structures is the early stage in a GTTM, an extremely accurate analyzer will make it possible to improve the performance of all later analyzers.

We previously constructed several analyzers, such as the automatic time-span tree analyzer (ATTA) [2] and fully automatic time-span tree analyzer (FATTA) [3], that enable us to detect local grouping boundaries. However, the performance of these analyzers is inadequate in that musicologists have to correct the boundaries due to numerous errors.

Our deepGTTM-I&II is based on a deep learning technique [4] to improve the performance of detecting local grouping boundaries and generating metrical structures of music pieces. Unsupervised training in the deep learning of deep-layered networks called pre-training helps in supervised training, which is called fine-tuning [5].

Our goal was to develop a GTTM analyzer that enables the output of the results obtained from analysis that were the same as those obtained by musicologists based on deep learning by learning the analysis results obtained by the musicologists. We had to consider the following three issues in constructing this analyzer.

Multi-task learning

A model or network in a simple learning task estimates the label from an input feature vector. However, local grouping boundaries can be found in many note transitions. Also, a strong beat can be found in many beat positions. Therefore, we consider a single learning task for estimating whether one note transition can be a boundary (a strong beat).

A problem in detecting local grouping boundaries or strong beats can then be solved using multi-task learning.

Subsection 4.3 explains multi-task learning by using deep learning.

Large-scale training data

Large-scale training data are needed to train a deep-layered network, and labels are not needed in pre-training the network. Therefore, we collected 15,000 pieces of music formatted in musicXML from Web pages that were introduced in the MusicXML page of MakeMusic Inc citeMakeMusic. We needed labeled data to fine-tune the network. Although we had 300 pieces with labels in the GTTM database [7], this number was too small to enable the network to learn.

Subsection 4.1 explains how we collected the data and how we got the network to learn effectively with a small dataset.

GTTM rules

A GTTM consists of multiple rules, and a note transition that is applied to many rules tends to be a local grouping boundary in the analysis of local grouping boundaries. Similarly, a beat that is applied to many rules tends

to be a strong beat in the analysis of the metrical structure. As a result of analysis by musicologists, 300 pieces of music in the GTTM database were not only labeled with local grouping boundaries and metrical structures but also labeled with applied positions of preference rules (PRs). Therefore, the applied positions of PRs were helpful clues in detecting local grouping boundaries and strong beats.
Subsection 4.3 explains how the network learned with the PRs.

Sequential vs. recurrent models
There are two types of models, i.e., recurrent and sequential, that can be used for GTTM analysis. The recurrent neural network provides recurrent models, which are suitable for analyzing a metrical structure in which cyclical change results in strong and weak beats. However, the recurrent neural network is difficult to train, and training time is very long. On the other hand, sequential models, such as deep belief networks (DBNs), are not suitable for detecting the repetition of strong beats. However, the deepGTTM-I DBN of our analyzer is very simple and performs well in detecting the local grouping boundary of a GTTM.
Therefore, we chose two DBNs for GTTM analysis. Subsection 4.2 explains how these DBNs are trained for analyzing metrical structures.

Hierarchical metrical structure
A hierarchical metrical structure is generated by iterating the choice of the next-level structure. The next level structure is recursively generated using the previous structure. However, when we use learning with a single standard network of deep learning, it is difficult to lean a higher-level structure because many network representations are used for learning a lower-level structure.
Subsection 4.2 explains how to learn a higher level structure.

The results obtained from an experiment indicate that our GTTMI&II involving multi-task learning using the deep learning technique outperformed previous GTTM analyzers.

The paper is organized as follows. Section 2 describes related work and Section 3 explains our analyzer called deepGTTM-I&II. Section 4 explains how we evaluated the performance of deepGTTM-I&II, and Section 5 concludes with a summary and overview of future work.

2 Related Work

We now briefly look back through the history of cognitive music theory. The imprecation realization model (IRM) proposed by Narmour abstracts and expresses music according to symbol sequences from information from a score [8, 9]. Recently, the IRM has been implemented on computers, and its chain structures can be obtained from a score [10]. The *Schenkerian* analysis analyzes deeper structures called "Urline" and "Ursatz" from the music surface [11]. Short segments of music can be analyzed through Schenkerian analysis on a computer [12]. There is another approach that constructs a music theory for adopting computer implementation [13, 14].

We consider a GTTM to be the most promising of the many theories that have been proposed [8, 9, 11, 13, 15] in terms of its ability to formalize musical knowledge because a GTTM captures the aspects of musical phenomena based on the Gestalt occurring in music and is presented with relatively rigid rules. The main advantage of analysis by using a GTTM is that it can acquire tree structures called time-span and prolongation trees.

We have been constructing both analyzers and an application of a GTTM for more than a decade (Fig. 1) citeHamanaka 2016. The horizontal axis in Fig. 1 indicates years. The analyzers we developed are above the timeline.

2.1 Analyzers for GTTM based on full parameterization

We first constructed a grouping structure analyzer and metrical structure analyzer (Figs. 1a and b). We developed the ATTA (Fig. 1c) [2] by integrating a grouping structure analyzer and a metrical analyzer. We extended the GTTM by full externalization and parameterization and proposed a machine-executable extension of the GTTM, exGTTM. We implemented exGTTM on a computer that we call the ATTA . The ATTA has 46 adjusted parameters to control the strength of each rule. The ATTA we developed enables us to control the priority of rules, which enables us to obtain extremely accurate groupings and metrical structures. However, we need musical knowledge that musicologists have to properly tune the parameters.

The FATTA [3] (Fig. 1d) does not have to tune the parameters because it automatically calculates the stability of structures and optimizes the parameters so that the structures would be stable. It achieved excellent analysis results for metrical structures, but the results for grouping structures and time-span trees were unacceptable.

We constructed an interactive GTTM analyzer [22](Fig. 1e) that enables seamless changes in the automatic analysis and manual editing processes because it was difficult to construct an analyzer that could output analysis results in the same way as musicologists. The interactive GTTM analyzer is still used to collect GTTM analysis data, and anyone can download and use it for free [23].

However, all these analyzers [2, 3, 22, 23] have problems. Musical knowledge is required for the ATTA to tune the parameters, and the FATTA performs poorly.

2.2 Analyzers for GTTM based on statistical learning

The σGTTM analyzer [24] (Fig. 1f) enables us to automatically detect local grouping boundaries by using a decision tree. Although σGTTM performed better than the FATTA, it was worse than the ATTA after the ATTA parameters had been tuned.

The σGTTMII analyzer [25] (Fig. 1g) involves clustering steps for learning the decision tree and outperforms the ATTA if we can manually select the best decision tree. Although σGTTMII performed the best in detecting grouping boundaries, it was difficult to select the proper decision tree without musical knowledge.

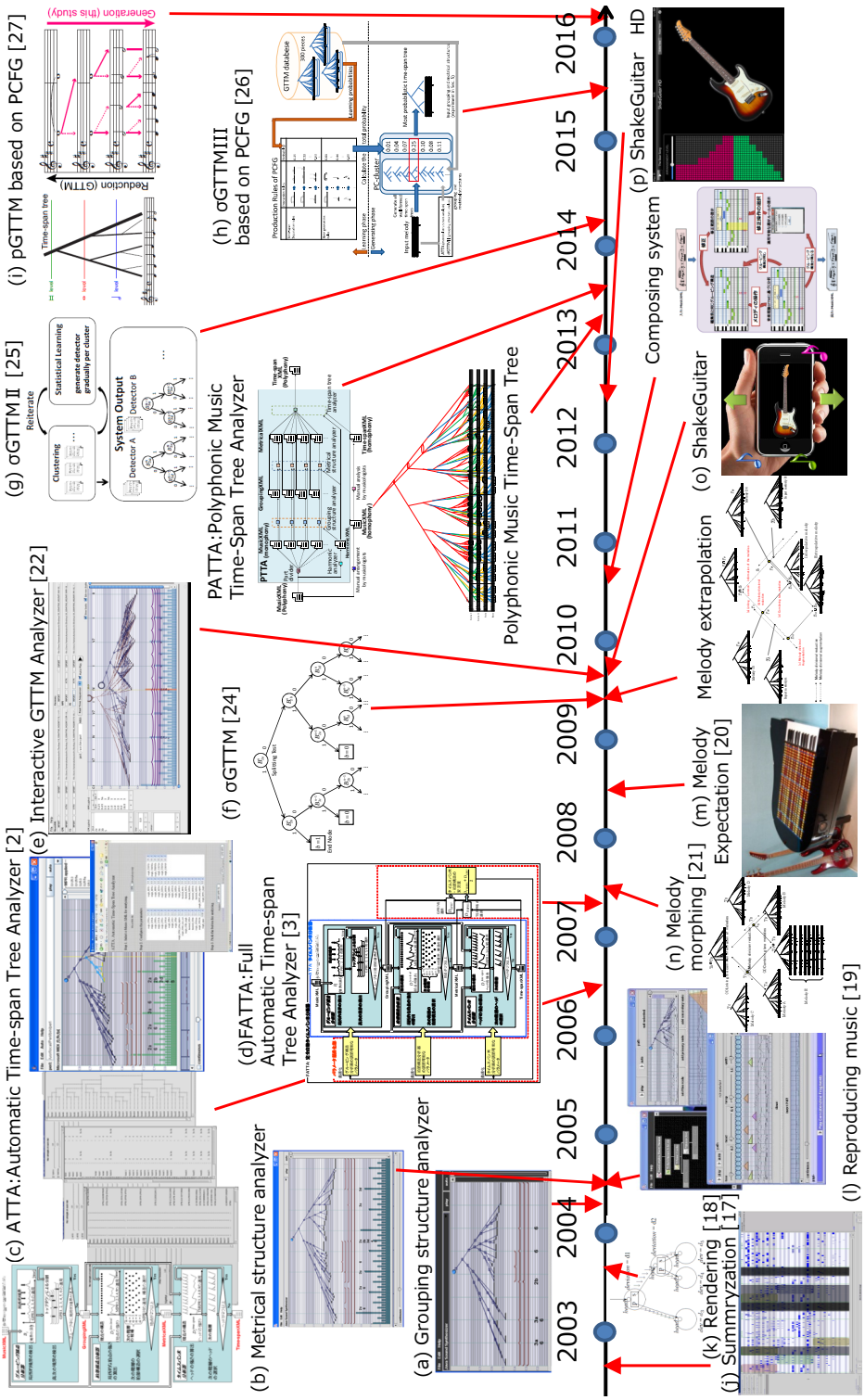

Fig. 1. Related work on analysis and application systems for GTTM

The σGTTMIII analyzer [26] (Fig. 1h) enables us to automatically analyze time-span trees by learning with a time-span tree of 300 pieces of music from the GTTM database [7] based on probabilistic context-free grammar (PCFG). The σGTTMIII analyzer performed the best in acquiring time-span trees. The pGTTM anlayzer [27] (Fig. 1i) also uses PCFG, and we used it to attempt unsupervised learning. The main advantages of σGTTMIII and pGTTM are that they can learn the contexts in difference hierarchies of the structures (e.g., beats were important in the leaves of time-span trees, or chords were important near the roots of the trees.).

However, all these analyzers [24–27] have problems with regard to detecting local grouping boundaries. The σGTTM III and pGTTM analyzers are focused on acquiring time-span trees and cannot acquire local grouping boundaries. Musical knowledge is required for σGTTM II to select the decision tree. Since σGTTM and σGTTM II use rules that musicologists applied, they cannot work as standalone analyzers. For example, information on parallel phrases is needed when detecting local grouping boundaries because parallel phrases create parallel structures in a GTTM. However, σGTTM and σGTTM II do not have processes for acquiring parallel phrases.

In light of this, we introduce a deep learning analyzer for detecting the local grouping boundaries and generating hierarchical metrical structures of music pieces based on a GTTM.

2.3 Application systems by using analysis results of GTTM

We constructed applications systems, which are given under the time-line in Fig. 1, to use the results from GTTM analysis. A time-span or prolongation tree provides a summarization of a piece of music, which can be used as the representation of an abstraction, resulting in a music retrieval system [17] (Fig. 1j). It can also be used for performance rendering [18] (Fig. 1k) and reproducing music [19](Fig. 1l). The time-span tree can also be used for melody prediction [20](Fig. 1m) and melody morphing [21](Fig. 1n). Figures 1o and p illustrate a demonstration system for the melody morphing method that changes the morphing level of each half bar by using the values from the accelerometer in the iPad/iPhone/iPod Touch. When the user stops moving the iPhone/iPod Touch, the unit plays the backing melody of "The Other Day, I Met a Bear (The Bear Song)". When the user shakes it vigorously, it plays heavy soloing. When the user shakes it slowly, it plays a morphed melody between the backing and heavy soloing.

These systems currently need a time-span tree analyzed by musicologists because our analyzers do not perform optimally.

2.4 Melody segmentation

Because conventional approches of melody segmentation, such as the Grouper of the Melisma Music Analyzer by Temperley [28] and the local boundary detection model (LBDM) by Cambouropoulos [29], require the user to make manual

adjustments to the parameters, they are not completely automatic. Although Temperley [30] has also used a probabilistic model, it has not been applied to melody segmentation. The unsupervised learning model (IDyOM) proposed by Pearce et al. makes no use of the rules of music theory with regard to melodic phrases, and it has performed as well as Grouper and LBDM [31]. However, as deepGTTM-I&II statistically and collectively learn all the rules for the grouping structure analysis of a GTTM, we expect that deepGTTM-I&II will perform better than an analyzer that only uses statistical learning.

2.5 Beat tracking

The metrical structure analysis in a GTTM is a kind of beat tracking. Current methods based on beat tracking [32–35] can only acquire the hierarchical metrical structure in a measure because they do not take into account larger metrical structures such as two and four measures.

3 GTTM and Its Implementation Problems

Figure 2 shows local grouping boundaries, a grouping structure, metrical structure, timespan tree, and prolongational tree.

The grouping structure is intended to formalize the intuitive belief that tonal music is organized into groups that are in turn composed of subgroups. These groups are presented graphically as several levels of arcs below a music staff.

The metrical structure describes the rhythmical hierarchy of a piece of music by identifying the position of strong beats at the levels of a quarter note, half note, measure, two measures, four measures, and so on. Strong beats are illustrated as several levels of dots below the music staff.

3.1 Preference rules

There are two types of rules in a GTTM, i.e., "well-formedness" and "preference". Well-formedness rules (WFRs) are necessary for the assignment of a structure and restrictions on the structure. When more than one structure satisfies the WFRs, the PRs indicate the superiority of one structure over another.

There are seven grouping PRs (GPRs) (**GPRs**): **GPR1** (alternative form), **GPR2** (proximity), **GPR3** (change), **GPR4** (intensification), **GPR5** (symmetry), **GPR6** (parallelism), and **GPR7** (time-span and prolongational stability). The **GPR2** has two cases: (a) (slur/rest) and (b) (attack-point). The **GPR3** has four cases: (a) (register), (b) (dynamics), (c) (articulation), and (d) (length).

There are ten metrical PRs (MPRs) (**MPRs**): **MPR1** (parallelism), **MPR2** (strong beat early), **MPR3** (event), **MPR4** (stress), **MPR5** (length), **MPR6** (bass), **MPR7** (cadence), **MPR8** (suspension), **MPR9**(time-span interaction), and **MPR10** (binary regularity). The **MPR5** has six cases: (a) pitch-event, (b) dynamics, (c) slur, (d) articulation, (e) repeated pitches, and (f) harmony.

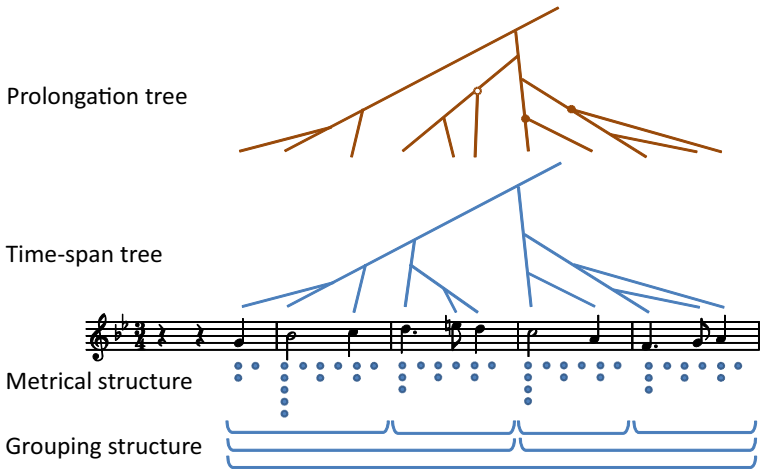

Fig. 2. Grouping structure, metrical structure, time-span tree, and prolongation tree

3.2 Conflict between rules

Because there is no strict order for applying PRs, a conflict between rules often occurs when applying PRs, which results in ambiguities in analysis.

Figure 3(a) outlines a simple example of the conflict between **GPR2b** (attack-point) and **GPR3a** (register). **GPR2b** states that a relatively greater interval of time between attack points initiates a grouping boundary. The **GPR3a** states that a relatively greater difference in pitch between smaller neighboring intervals initiates a grouping boundary. Because **GPR1** (alternative form) has strong preference for note 3 alone not forming a group, a boundary cannot be perceived at both 2-3 and 3-4.

Figure 3(b) shows an example of the conflict between **MPRs 5c** and **5a**. The **MPR5c** states that a relatively long slur results in a strong beat, and **MPR5a** states that a relatively long pitch-event results in a strong beat. Because metrical WFR 3(**MWFR3**) states that strong beats are spaced either two or three beats apart, a strong beat cannot be perceived at both onsets of the first and second notes.

We expect to learn the rule application and priority of rules by inputting a whole song with labels of the applied rules to a deep-layered network.

3.3 Ambiguous rule definition

Some rules in a GTTM are expressed with ambiguous terms. The **GPR4** and **MPR5** are defined as follows as examples.

> The **GPR4 (Intensification)**, where the effects selected by GPRs 2 and 3 are relatively more pronounced, a larger-level group boundary may be placed.

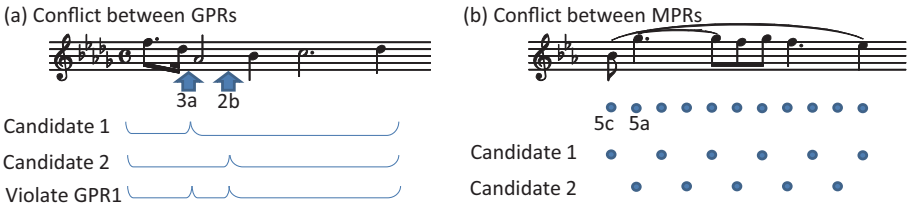

Fig. 3. Examples of conflict between PRs

The **MPR5 (Length)** has preference for a metrical structure in which a relatively strong beat occurs at the inception of either

a. a relatively long pitch-event,
b. relatively long duration of a dynamic,
c. relatively long slur,
d. relatively long pattern of articulation,
e. relatively long duration of a pitch in the relevant levels of the time-span reduction, or
f. relatively long duration of a harmony in the relevant levels of the time-span reduction (harmonic rhythm).

The term "relatively" in this sense is ambiguous. The sentence also contains the phrase "more pronounced", but the comparison is unclear. Another example is that a GTTM has rules for selecting proper structures when discovering similar melodies (called parallelism) but does not define similarity. The **GPR6** and **MPR1** are defined as follows as examples.

The **GPR6 (parallelism)**, when two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups.
The **MPR1 (parallelism)**, when two or more groups or parts of groups can be construed as parallel, they preferably receive a parallel metrical structure.

3.4 Context dependency

To solve the problems discussed in Subsections 3.2 and 3.3, we proposed the machine executable extension of GTTM (exGTTM) and ATTA [2]. Figure 4 gives an example of an application of **MPR4**, **5a**, **5b**, and **5c** in the exGTTM and ATTA. By configuring the threshold parameters $T^j(j = 4, 5a, 5b, \textit{and } 5c)$, we can control whether each rule is applicable. However, proper values of the parameter depend on the piece of music and level of hierarchy in the metrical structure. Therefore, automatic estimation of proper values of the parameters is difficult.

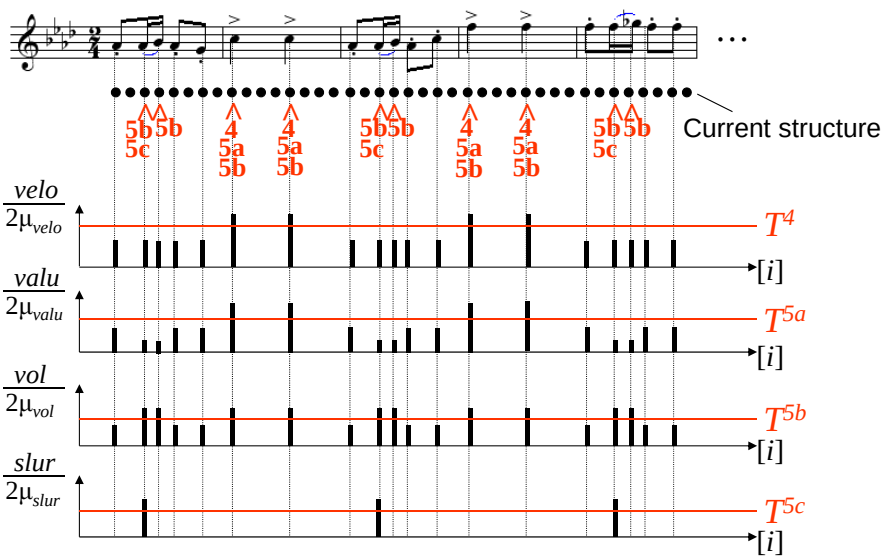

Fig. 4. Application of MPR4, 5a, 5b, and 5c in ATTA

3.5 Less precise explanation of feedback link

A GTTM has various feedback links from higher-level structures to lower-level ones. For example, **MPR9** is defined as follows.

> The **MPR9 (Time-span interaction)** has preference for a metrical analysis that minimizes conflict in the time-span reduction.

However, no detailed description and only a few examples are given. Other feedback links in the GTTM rules are not explicit. For example, analyzing the results of a time-span tree strongly affects the interpretation of chord progression, and various rules are related to chord progression, e.g., **MPR7 (Cadence)** requires a metrical structure in which cadences are metrically stable.

For complete implementation of a GTTM based on deep learning, we have to introduce the feedback link by using the recurrent neural network; however, we do not focus on the feedback link in this paper.

4 deepGTTM-I&II: Local Grouping Boundary and Metrical Strucutre Analyzer based on Deep Learning

We introduced deep learning into deepGTTM-I&II to analyze the structure of a GTTM and solve the problems described in Subsections 3.2, 3.3 and 3.4. There are two main advantages of introducing deep learning.

Learning rule applications

We constructed a deep-layered network that can output whether each rule

is applicable on each note transition by learning the relationship between the scores and positions of applied grouping PRs with the deep learning technique.
Previous analysis systems based on GTTM were constructed by a human researcher or programmer. As described in Subsection 3.3, some rules in a GTTM are very ambiguous and the implementations of these rules might differ depending on the person.
However, deepGTTM-I&II is a learning based analyzer, where its quality depends on the training data and trained network.

Learning priority of rules

The FATTA only determine the priority of rules from the stability of the structure. The σGTTM and σGTTMII analyzers only determine the priority of rules from applied rules. They do not work well because the priority of rules depends on the context of a piece of music.
The input of the network in deepGTTM-I&II, on the other hand, is the score and the network learns the priority of the rules as the weight and bias of the network based on the context of the score.

This section describes how we analyzed the local grouping boundaries and metrical structure by using deep learning.

4.1 Datasets for training

Three types of datasets were used to train the network, i.e., a non-labeled dataset for pre-training, half-labeled dataset, and labeled dataset for fine-tuning (Fig. 5).

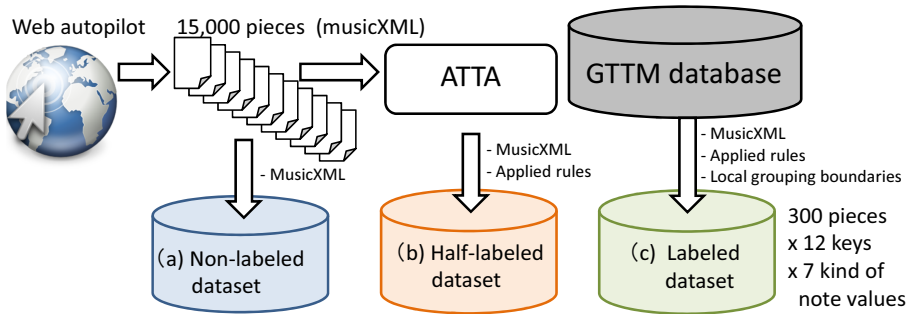

Fig. 5. Non-labeled, half-labeled, and labeled datasets

Non-labeled dataset

The network in pre-training learned the features of the music. A large-scale dataset with no labels was needed. Therefore, we collected 15,000 pieces of music formatted in musicXML from Web pages that were introduced on the musicXML page of MakeMusic Inc. [6] (Fig. 5a). The musicXMLs were downloaded in the following three steps.

a. Web autopilot script made a list of urls that probably downloaded musicXMLs in five links from the musicXML page of MakeMusic Inc.
b. The files in the url list were downloaded after they had been omitted because they were clearly not musicXML.
c. All the downloaded files were opened using the script, and files that were not musicXML were deleted.

Half-labeled dataset

The network in fine-tuning learned with the labeled dataset. We had 300 pieces of music with a labeled dataset in the GTTM database, which included musicXML with positions of local grouping boundaries, and positions to which the GPRs were applied. However, 300 pieces are insufficient for deep learning.

Consequently, we constructed a half-labeled dataset. We automatically added the labels of thirteen applied rules of **GPR2a**, **2b**, **3a**, **3b**, **3c**, **3d**, and **MPR2**, **3**, **4**, **5a**, **5b**, **5c**, and **5d**, because these rules could be uniquely applied as a score. We used the ATTA to add labels to these rules (Fig. 5b). With the ATTA, the strength of the beat dependent on each MPR can be expressed as ${D_i}^j (j = 2, 3, 4, 5a, 5b, 5c, \ and \ 5d, \ 0 \leq {D_i}^j \leq 1)$. For example, **MPR4** is defined in a GTTM as follows.

> The **MPR4 (Event)** has preference for a metrical structure in which beats of level L_i that are stressed are strong beats of L_i.

We formalized ${D_i}^4$ as follows.

$$ {D_i}^4 = \begin{cases} 1 & velo_i > 2 \times \mu_{velo} \times T^4 \\ 0 & else, \end{cases} \tag{1} $$

where $velo_i$ is the velocity of a note from beat i, μ_{velo} is the average of $velo_i$, and T^j $(0 \leq T^j \leq 1)$ are the threshold parameters to control those that determine whether the rules are applicable $({D_i}^j = 1)$ $({D_i}^j = 0)$. We used 1 as the threshold parameter value $(T^j = 1, \ where \ j = 2, 3, 4, 5a, 5b, 5c, \ and \ 5d)$.

Labeled dataset

We collected 300 pieces of 8-bar-long, monophonic, classical music and asked people with expertise in musicology to analyze them manually with faithful regard to the MPRs. These manually produced results were cross-checked by three other experts.

We artificially increased the labeled dataset because 300 pieces of music in the GTTM database were insufficient for training a deep-layered network. First, we transposed the pieces for all 12 keys. We then changed the length of the note values to two times, four times, eight times, half time, quarter time, and eighth time. Thus, the total labeled dataset had 25,200 (= 300x12x7) pieces of music (Fig. 5c).

4.2 Deep belief networks

We used deep belief networks (DBN) for detecting local grouping boundaries and generating metrical structures.

Figure 6 outlines the structure of this DBN, which we call deepGTTM-I , for detecting local grouping boundaries. The inputs of deepGTTM-I are the onset time, offset time, pitch, and velocity of note sequences from musicXML There are 11 outputs of deepGTTM-I to enable multi-tasking learning, i.e., ten GPRs (**GPR2a**, **2b**, **3a**, **3b**, **3c**, **4**, **5**, **6**, and **7**) and a local grouping boundary.

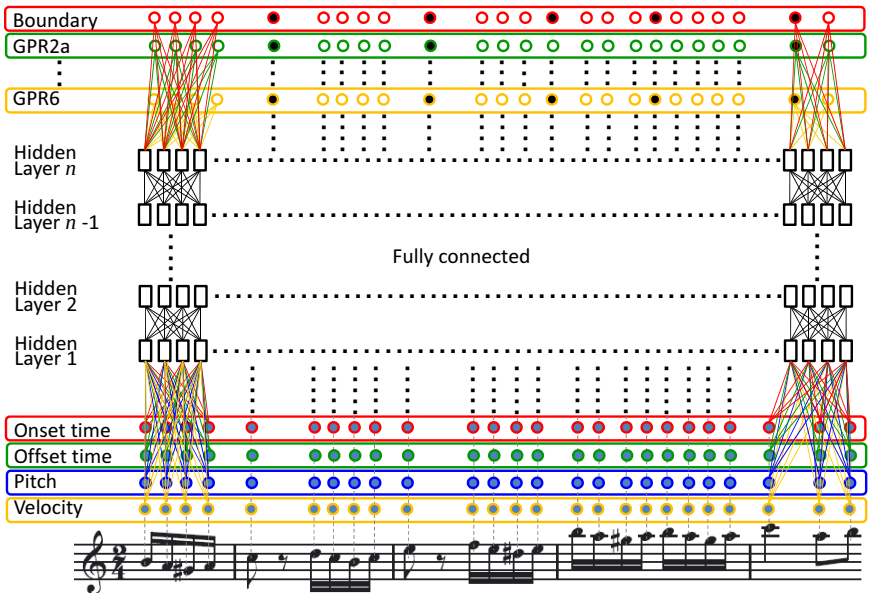

Fig. 6. DBN for detecting local grouping boundaries

Figure 7 outlines the structure of the DBN we call deepGTTM-II to generate a metrical structure . The inputs of deepGTTM-II are the onset time, offset time, pitch, and velocity, and grouping structure manually analyzed by musicologists. Each hierarchical level of the grouping structure is separately inputted by a note neighboring the grouping boundary as 1; otherwise, 0. There are eight outputs of deepGTTM-II to enable multi-tasking learning in each hierarchical level of the metrical structure, i.e., seven MPRs (**MPR2**, **3**, **4**, **5a**, **5b**, **5c**, and **5d**) and one level of the metrical structure. Individual outputs have two units, e.g., rules that were not applicable (=0) and rules that were applicable (=1), or weak beats (=0) and strong beats (=1).

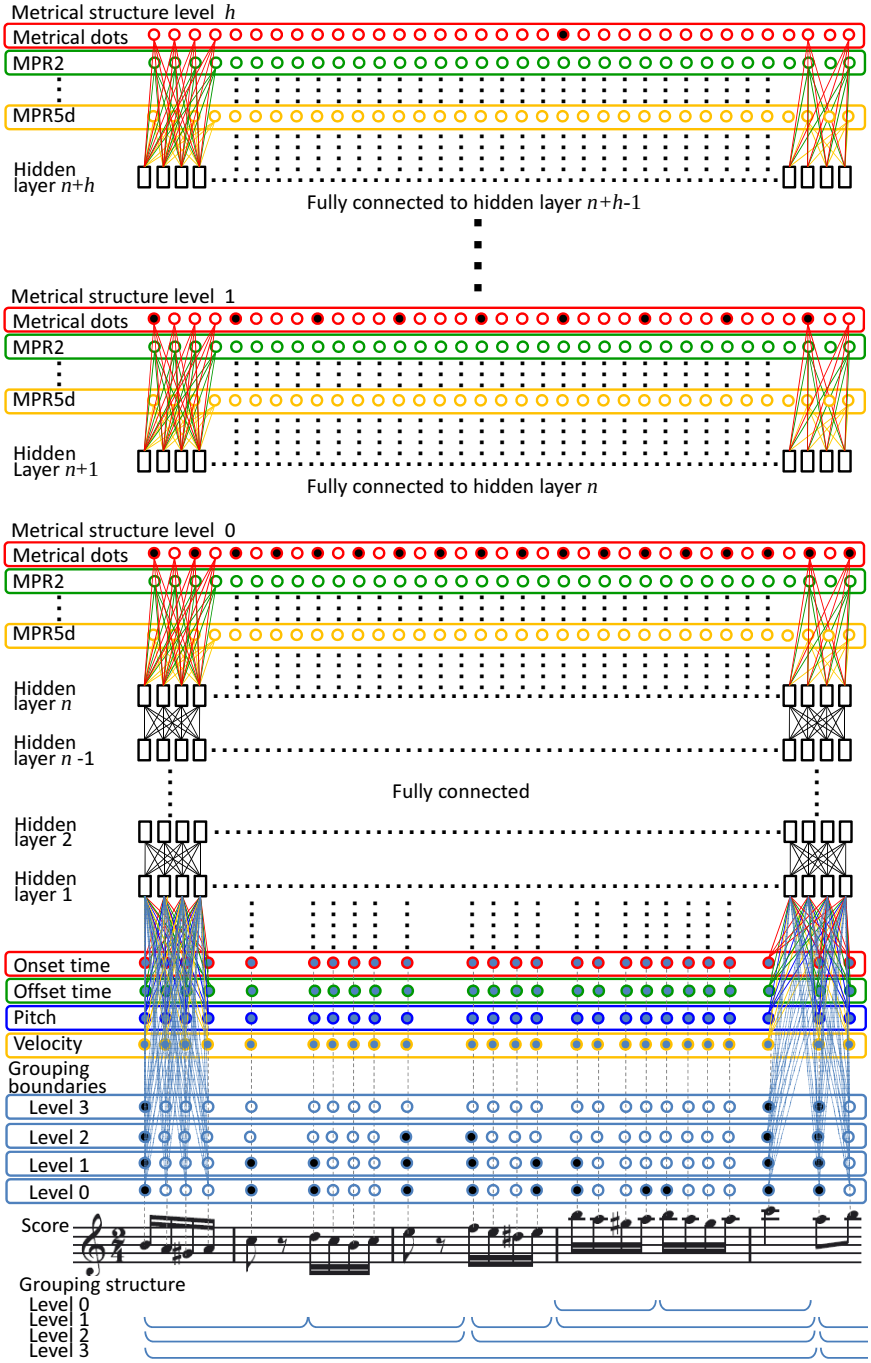

Fig. 7. DBN for generating a metrical structure

A metrical structure consists of hierarchical levels, and we added one hidden layer to generate the next structure level. We used logistic regression to connect the final hidden layer (n,n+1,..., n+h) and outputs. All outputs shared the hidden layer from 1 to the final hidden layer. The network was learned in the four steps below. The order of music pieces was changed at every epoch in all steps.

4.3 Multi-dimensional multi-task learning

Our deepGTTM-I&II consists of a very complex network. The fine-tuning of local grouping boundaries and one level of a metrical structure involves multi-task learning. The fine-tuning of each PR also involved multi-task learning. Therefore, the fine-tuning of PRs involves multi-dimensional multi-task learning (Fig. 8).

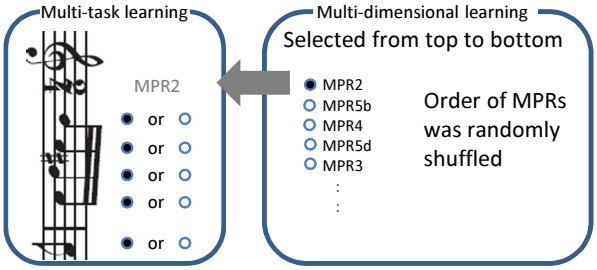

Fig. 8. Multi-dimensional multi-task learning

Multi-task learning The processing flow for the multi-task learning of a GPR or local grouping boundaries involves the following four steps.

Step 1: The order of the pieces of training data is randomly shuffled and a piece is selected from top to bottom.
Step 2: The note transition of the selected piece is randomly shuffled and a note transition is selected from top to bottom.
Step 3: Back propagation from output to input is carried out based on whether the note transition had a boundary or the rule was applied (=1) or not (=0).
Step 4: The next note transition or the next piece in steps 1 and 2 is repeated.

The processing flow for the multi-task learning of an MPR or metrical dots involved four steps.

Step 1: The order of the music pieces of training data is randomly shuffled and a piece is selected from top to bottom.

Step 2: The beat position of the selected piece is randomly shuffled and a beat position is selected from top to bottom.
Step 3: Back propagation from output to input is carried out based on whether the beat position had a strong beat or the rule was applied (=1) or was not (=0).
Step 4: The next beat position or the next piece in steps 2 and 3 is repeated.

Multidimensional multi-task learning The processing flow for the multidimensional multi-task learning of PRs involves the following three steps.

Step 1: The order of PRs is randomly shuffled and a rule is selected from top to bottom.
Step 2: Multi-task learning of the selected PR is carried out.
Step 3: The next rules in step 1 are repeated.

5 Experimental Results

We evaluated the F_{measure} of deepGTTM-I&II by using 100 music pieces from the GTTM database; the remaining 200 pieces were used to train the network. The F_{measure} is given by the weighted harmonic mean of precision P (proportion of selected dots that are correct) and recall R (proportion of correct dots that were identified).

$$F \text{ measure} = 2 \times \frac{P \times R}{P + R} \tag{2}$$

Tables 1 and 2 summarize the results of deepGTTM-I and deepGTTM-II, respectively, for a network that had 11 layers with 3000 units.

The results in the tables indicate that deepGTTM-I&II outperformed previous analyzers in terms of the F-measure.

The ATTA has adjustable parameters and its performance changes depending on the parameters. For the default parameter, we used the middle value in the parameter range [2]. The σGTTMII analyzer selected the decision tree and the performance changes on the selected tree. The performances of the ATTA and σGTTMII changed depending on the parameters or decision trees. Table 1 indicates the best performance was achieved by manual editing. However, the FATTA, σGTTM, and deepGTTM-I have no parameters for editing.

These results show that deepGTTM-I&II performed extremely robustly.

6 Conclusion

We developed an analyzer for detecting local grouping boundaries and generating a metrical structure called deepGTTM-I&II that is based on deep learning. The following three points are the main results of this study.

Table 1. Performances of deepGTTM-I, ATTA, σGTTM

	Precision P	Recall R	F measure
ATTA with manual editing of parameters	0.74	0.44	0.55
σGTTM	0.47	0.74	0.57
σGTTM with manual selection of decision tree	0.68	0.92	0.78
deepGTTM-I	0.78	0.81	0.80

Table 2. Performances of deepGTTM-II, ATTA, and FATTA

Melodies	deepGTTM-II	ATTA (Default prameters)	ATTA (Configured parameters)	FATTA
1. Grande Valse Brillante	0.94	0.88	0.93	0.88
2. Moments Musicaux	1.00	0.95	1.00	1.00
3. Trukish March	0.98	0.91	0.96	0.96
4. Anitras Tanz	0.90	0.82	0.86	0.82
5 Valse du Petit Chien	0.99	0.87	0.92	0.95
	:	:	:	:
Total (100 melodies)	0.96	0.84	0.90	0.88

Music analyzer based on deep learning

It has been revealed that deep learning is strong for various tasks. We demonstrated that deep leaning is also strong for music analysis. We will attempt to implement other music theories based on deep learning. Although we collected 300 pieces of music and analyzed the results of a GTTM by musicologists, the labeled dataset of these pieces was not sufficient for learning a deep-layered network. We therefore used our previous GTTM analyzer called ATTA to prepare three types of datasets, non-labeled, half-labeled, and labeled, to learn the network.

High-accuracy GTTM analyzer without manual editing

Previous GTTM analyzers, such as the ATTA and σGTTM, require manual editing; otherwise, the performance will be much worse. The F_{measure}s of GTTM analyzers without manual editing, such as the FATTA, σGTTM, σGTTMIII, and pGTTM, are too low (under 0.8). However, deepGTTM-I&II exhibited extremely high performance, which indicates the possibility of practical use in GTTM applications [17–21]. We plan to implement the entire GTTM analysis process based on deep learning.

Multi-dimensional multi-task learning

We constructed a multi-dimensional multi-task learning analyzer that efficiently learns the grouping boundaries and hierarchical level of the metrical structure and PRs by sharing the network. Multi-dimensional multi-task learning is expected to be applied to other data that have a hierarchy and time series such as film [36] and discussion [37]. After a network that had 11

layers with 3000 units had been learned, the deepGTTM-I&II outperformed the previously developed analyzers in terms of the F_{measure}.

This work was one step in implementing a GTTM based on deep learning. The remaining steps are to implement time-span reduction analysis and prolongational reduction analysis of a GTTM based on deep learning. The following two problems remain. One is generating tree structures because time-span and prolongation tree structures are more complex than local grouping boundaries or a hierarchical metrical structure. The other problem is the lack of training samples because there are many combinations of tree structures, and an unlearned sample sometimes appears in test data. We will attempt to solve these problems and make it possible to construct a complete GTTM system based on deep learning.

At the current stage, we are not able to understand the details on why deep learning works extremely well for metrical analysis in a GTTM. Thus, we also plan to analyze a network after a metrical structure is learned.

Acknowledgments This work was supported by JSPS KAKENHI Grant Number 25700036, 16H01744, 23500145.

References

1. Lerdahl, F., and Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge (1983)
2. Hamanaka, M., Hirata, K., and Tojo, S.: Implementing 'A Generative Theory of Tonal Music', Journal of New Music Research, 35:4, 249–277 (2006)
3. Hamanaka, M., Hirata, K., and Tojo, S.: FATTA: Full Automatic Time-span Tree Analyzer, In: Proceedings of the 2007 International Computer Music Conference (ICMC2007), pp. 153–156 (2007)
4. Hinton, G. E., Osindero, S., and Teh Y.-W.: A fast learning algorithm for deep belief nets, Neural Computation,18:7, 1527–1554 (2006)
5. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S.: Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research, vol. 11, 625–660 (2010)
6. MakeMusic Inc.: Finale, `http://www.finalemusic.com/`, see at 2017-1-4.
7. Hamanaka, M., Hirata, K., and Tojo, S.: Music Structural Analysis Database based on GTTM, In: Proceedings of the 2014 International Society for Music Information Retrieval Conference (ISMIR2014), pp. 325–330 (2014)
8. Narmour E: The Analysis and Cognition of Basic Melodic Structure. University of ChicagoPress (1990).
9. Narmour E: The Analysis and Cognition of Melodic Complexity. The University of ChicagoPress (1992)
10. Yazawa, S. and Hamanaka, M. and Utsuro, T.: Melody Generation System based on a Theory of Melody Sequences, In: Proceedings of ICAICTA2014, pp. 347–352 (2014)
11. Schenker, H. Der frei Satz. Vienna: Universal Edition, 1935. Published in English as Free Composition, translated and edited by E. Oster, New York: Longman (1979)

12. Marsden, A.: Software for Schenkerian Analysis, In: Proceeding of International Computer Music Conference (ICMC2011), pp. 673–676 (2011)
13. Temperley, D.: The Congnition of Basic Musical Structures. MIT Press, Cambridge (2004)
14. Lerdahl, F.: Tonal Pitch Space, Oxford University Press (2001)
15. Cooper, G., and Meyer, L. B.: The Rhythmic Structure of Music. The University of Chicago Press (1960)
16. Hamanaka ,M., Hirata, K., and Tojo, S.: Implementing Methods for Analysing Music Based on Lerdahl and Jackendoff's Generative Theory of Tonal Music, Computational Music Analysis, 221–249, Springer (2016)
17. Hirata, K., and Matsuda, S.: Interactive Music Summarization based on Generative Theory of Tonal Music. Journal of New Music Research, 32:2, 165–177 (2003)
18. Hirata, K., and Hiraga, R.: Ha-Hi-Hun plays Chopin's Etude, Working Notes of IJCAI-03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest, pp. 72–73 (2003)
19. Hirata, K., and Matsuda, S.: Annotated Music for Retrieval, Reproduction, and Sharing, In: Proceeding of International Computer Music Conference (ICMC2004), pp. 584–587 (2004)
20. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Expectation Method based on GTTM and TPS, In: Proceeding of the 2008 International Society for Music Information Retrieval Conference (ISMIR2008), pp. 107–112 (2008)
21. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Morphing Method based on GTTM, In: Proceeding of the 2008 International Computer Music Conference (ICMC2008), pp. 155–158 (2008)
22. Hamanaka, M., Hirata, K., and Tojo, S.: Interactive GTTM Analyzer, In: Proceedings of the 10th International Conference on Music Information Retrieval Conference (ISMIR2009), pp. 291-296 (2009)
23. Hamanaka, M.: Interactive GTTM Analyzer/GTTM Database, `http://gttm.jp`, seen on 2017-1-4.
24. Miura, Y., Hamanaka, M., Hirata, K., and Tojo, S.: Use of Decision Tree to Detect GTTM Group Boundaries, In: Proceedings of the 2009 International Computer Music Conference (ICMC2009), pp. 125–128 (2009)
25. Kanamori, K., and Hamanaka, M.: Method to Detect GTTM Local Grouping Boundarys based on Clustering and Statistical Learning, In: Proceedings of the 2014 International Computer Music Conference (ICMC2014), pp. 125–128 (2014)
26. Hamanaka, M., Hirata, K., and Tojo, S.: σGTTM III: Learning-based Time-span Tree Generator Based on PCFG, In: Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR 2015), pp. 303-317 (2015)
27. Nakamura, E., Hamanaka, M. Hirata, K., and Yoshii, K.: Tree-Structured Probabilistic Model of Monophonic Written Music Based on the Generative Theory of Tonal Music, In: Proceedings of The 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2016), pp. 276–280 (2016)
28. Temperley, D.: The Melisma Music Analyzer. `http://www.link.cs.cmu.edu/music-analysis/` (2003), seen on 2017-1-4.
29. Cambouropoulos, E. :The Local Boundary Detection Model (LBDM) and its application in the study of expressive timing, In: Proceedings of the International Computer Music Conference (ICMC2001), pp. 290-293 (2001)
30. Temperley, D.: Music and Probability. Cambridge: The MIT Press (2007)

31. Pearce, M. T., Müllensiefen, D., and Wiggins, G. A.: A comparison of statistical and rule-based models of melodic segmentation, In: Proceedings of the International Conference on Music Information Retrieval (ISMIR2008), pp. 89-94 (2008)
32. Rosenthal, D.: Emulation of human rhythm perception, Computer Music Journal, 16:1, 64–76 (1992)
33. Goto, M.: An Audio-based Real-time Beat Tracking System for Music With or Without Drum-sounds, Journal of New Music Research, 30:2, 159–171 (2001)
34. Dixon, S.: Automatic Extraction of Tempo and Beat from Expressive Performance, Journal of New Music Research, 30:1, 39–58 (2001)
35. Davies, M. and Bock, S.: Evaluating the Evaluation Measures for Beat Tracking, In: Proceedings of the International Conference on Music Information Retrieval (ISMIR2014), pp. 637-642 (2014)
36. Takeuchi, S. and Hamanaka, M.: Structure of the film based on the music theory, in JSAI2014, 2014, 1K5-OS-07b-4 (in Japanese).
37. Oshima, T., Hamanaka, M., Hirata, K., Tojo, S. and Nagao, K.: Development of discussion structure editor for discussion mining based on muisc theory, ” in IPSJ SIG DCC, 2013, 7 pages (in Japanese).