deepGTTM-IV: Deep Learning Based Time-span Tree Analyzer of GTTM

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

[1] RIKEN
[2] Future University Hakodate
[3] Asia University
masatoshi.hamanaka@riken.jp

Abstract. This paper describes our development of a deep learning based time-span tree analyzer of the Generative Theory of Tonal Music (GTTM). Construction of a time-span tree analyzer has been attempted several times, but most previous analyzers performed very poorly, while those that performed relatively well required parameters to be manually adjusted. We previously proposed stepwise reduction for a time-span tree, which reduces the branches of the tree one by one, and confirmed that it can be learned by using the Transformer model. However, stepwise reduction could not obtain a time-span tree because it does not know to which notes the reduced notes were absorbed. Therefore, we improved the encoding for learning stepwise reduction and specified which notes are absorbed by which notes. We also propose a time-span tree acquisition algorithm that iterates stepwise reduction by representing the time-span tree as a matrix. As a result of experiments with 30 pieces, correct time-span trees were obtained for 29 pieces.

Keywords: Generative theory of tonal music (GTTM), time-span tree, melody reduction, Transformer model0

1 Introduction

We have developed a time-span tree analyzer that is based on the Generative Theory of Tonal Music (GTTM) by using deep learning called deepGTTM-IV. The GTTM was proposed by Leardahl and Jackendoff in 1983, and the time-span tree is a binary tree with each branch connected to each note [1].

Many time-span tree analyzers have been proposed, but most have many analytical errors [2–6]. The time-span tree analyzer that had the highest analytical performance required parameters to be manually adjusted [7].

The reason previous time-span tree analyzers performed insufficiently is that they analyzed in a bottom-up manner using only local information. [2–7]. Therefore, we considered learning the raw data of the entire piece by deep learning. Our deepGTTM-IV has four features.

Stepwise reduction: The Ground Truth data of a time-span tree is insufficient to directly learn the relationship between a piece and its time-span tree. To enable learning, we set the learning target as the process of reducing one note.

Branch priority: To make possible the stepwise reduction, the priority order of branches needs to be defined. The maximum time span is used as the branch priority.

Encoding: By encoding the score into text, stepwise reduction can be learned in the framework of automatic translation. This makes it possible to reduce notes at designated positions in a piece as if words are omitted in a sentence.

Time-span-tree matrix: The time-span tree has been handled in XML and Json formats, making coding difficult [8]. We made coding easier by expressing the information necessary for reduction (i.e., pitch, duration, time-span-tree shape, and branch priority) in a matrix.

We performed an experiment in which 270 items from a GTTM analysis corpus consisting of 300 pieces and their time-span trees were used to learn the Transformer model with the remaining 30 used for evaluation and found that our analyzer was able to obtain correct time-span trees for 29 out of 30 pieces. The remainder of the paper is as follows. Section 2 presents problems of time-span tree analysis based on deep learning, Section 3 describes the data for learning and evaluation, and Section 4 describes the implementation of the analyzer. Section 5 describes the experimental results, and Section 6 gives a summary and mentions future plans.

2 Problems of Time-span Tree Analysis based on Deep Learning

In GTTM analysis, the relationship between structurally important notes and other notes in a score is expressed by a binary tree called a time-span tree. The time-span tree in Fig. 1 is the result of analyzing Melody A on the basis of GTTM. Reduced melodies can be extracted by cutting this time-span tree with a horizontal line and omitting the notes connected below the line. In melody reduction with GTTM, decorative notes are absorbed by structurally important notes.

There are the following three problems in the deep learning of time-span tree analysis.

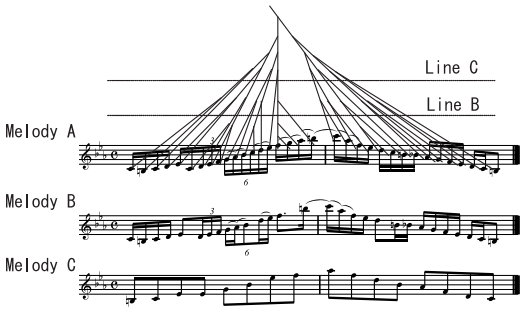

Fig. 1. Time-span tree

2.1 Low Number of Ground Truth Data

As ground truth data of the time-span tree, 300 classical melodies and their time-span trees are published in the GTTM database [8, **?**]. However, the number of datasets (300) is extremely small for learning deep neural networks (DNNs) [10]. In the case of a small number of pieces of learning data, over-fitting is inevitable, and an appropriate value cannot be output when unknown data is input.

In the time-span analysis by musicologists, the entire time-span tree cannot be acquired at once but is gradually analyzed from the bottom up. Therefore, the minimum process of analysis is set as one dataset, and then the number of datasets is increased. For example, if the DNN directly learns the relationship between a four-note melody and its time-span tree, the number of datasets is only one. On the other hand, if we consider the process of reducing one note to one dataset, the number of datasets will be three, as shown in Fig. 2(a).

The trained DNN estimates the melody consisting of $n-1$ notes that is reduced to one note when a melody consisting of n notes is input. A time-span tree for a melody consisting of four notes can be constructed by estimating four to three notes, three to two notes, and two notes to one note, and combining the results (Fig. 2(b)).

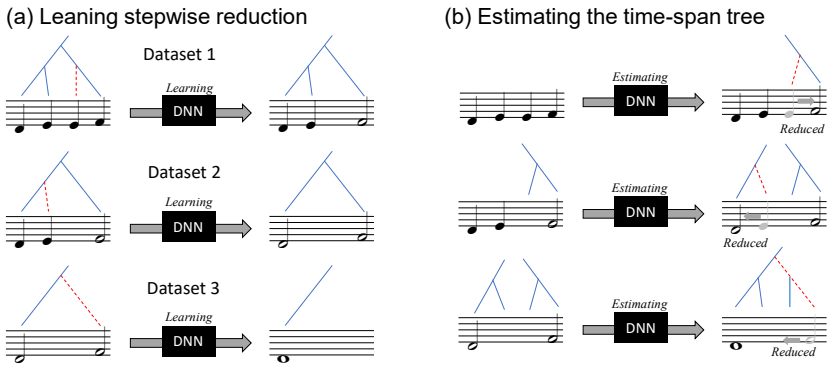

Fig. 2. Stepwise reduction

2.2 Ambiguity of Reduction Process

Time-span reduction removes decorative notes by pruning from the leaves at the tip of the tree, leaving only structurally important notes in the melody. To implement the stepwise reduction, the priority of branches must be obtained in a total order.

However, when it comes to GTTM itself, there are only a few examples of reduction using the time-span tree, and there is no detailed explanation on the reduction procedure [1]. For example, in Fig. 1, we can see two levels of reduction results, but it is not clear how many levels are actually necessary.

Marsden *et al*. [11] suggested a way to determine the salience of two note events (a and b), neither of which are descendants of the other. They proposed defining the salience of an event as the duration of the maximum of the time spans of the two children at the branching point when the event is generated, or where it is reduced.

In contrast, in this study, the DNN needs to learn the relationship before and after the reduction than it is to reduce the order of the notes to close to that of human cognition. We use a time-span tree leveled by the duration of the time span for a simple reduction order that it is easy for the DNN to learn [12].

2.3 Long Note Sequence

The previous time-span tree analyzers performed poorly because they analyzed in a bottom-up manner using only local information [2, 5–7]. In contrast, we propose using the entire note sequence before and after stepwise reduction for learning the DNN.

When a recurrent neural network (RNN) [13] or long short-term memory (LSTM) [14] is used as the DNN, the DNN can learn using note sequence, but when a long note sequence is input, the DNN forgets the beginning of it, and then the DNN cannot make use of the whole information of the note sequence.

The Transformer model [15] can learn and predict using the information of the entire note sequence. Moreover, the Transformer model has an additional layer of position information independently and uses the absolute position.

3 Data for deepGTTM-IV

This section describes the data for training the Transformer model. The Transformer model, which is an automatic translation tool, uses text for both input and output. Also, the Transformer model can learn the task of adding two values [16]. The duration of a note after reduction is the sum of the durations of the two notes before reduction, and we thought that this task could also be done with Transformer.

3.1 Learning and Evaluation Data

The preparation of the dataset for stepwise reduction is as follows. First, the priority of each branch of the time-span tree is evaluated on the basis of the duration of the maximum time span [12]. We refer to the longest temporal interval when a given pitch event becomes most salient as the maximum time span for the event. Next, stepwise reduction is applied to the least important note. A learning dataset of stepwise reduction is then created using the data before stepwise reduction as input data and the data after reduction as output data.

3.2 Encoding

Learning data are created from MusicXML and time-spanXML in the GTTM database. Since all melodies in the GTTM database are monophonic, the reduction method is limited to monophony. The notes in the melodies are made into a one-character string

with the pitch and duration concatenated. The pitch is represented as 12 types without distinguishing between different octaves. By multiplying by 4, the duration of most notes becomes an integer, but since there are melodies containing only a few triplets, quintuplets, sextuplets, and septuplets, the duration is rounded up to an integer. The placeholders "l" or "r" are inserted at positions where notes disappeared due to the reduction. The "l" (left) is inserted when the reduction is absorbed into the left note, and "r" (right) is inserted when it is absorbed into the right note. In our previous work, we were unable to reconstruct the time-span tree because we did not distinguish between "l" and "r" [12]. Figure 3 is an example of learning data.

Before reduction. → After reduction.

c14 c16 d30 **c14 c12** c16 d20 c16 . → c14 c16 d30 **c26 l** c16 d20 c16.
c14 c16 d30 c26 **c16 d20** c16 . → c14 c16 d30 c26 **r d36** c16.
c14 c16 d30 **c26 d36** c16 . → c14 c16 d30 **r d62** c16.
c14 c16 d30 d62 c16 . → **c30 l** d30 R2 d62 c16.
c30 d30 d62 c16 . → **c60 l** d62 c16.
c60 **d62 c16**. → c62 **r c78**.
c60 c78. → **r c138**.

Fig. 3. Learning data for melody reduction

As a result of preparing the datasets, 7362 stepwise reduction training datasets are generated from 270 music pieces from the GTTM database consisting of 300 pieces and 849 stepwise reduction evaluation datasets are generated from the remaining 30 pieces for evaluation.

3.3 Data Augmentation

The 7362 training datasets are not enough to train the Transformer model, so we carry out data augmentation. Each note is shifted 11 times by a semitone and the amount of training data is augmented by 12 times. The durations of notes are 2-16 times and rounded up to the nearest integer, then the amount of data is augmented by 16 times. Finally, we prepare 1,432,704 (= 7362 x 12 x 16) learning datasets.

4 Implementation of deepGTTM-IV

A time-span tree is obtained by iterating stepwise reduction. We expressed time-span trees in XML or Json, but they were difficult to handle with programs because of their deep hierarchical tree structure. Representing a time-span tree as a matrix makes melody reduction easier to implement in a program.

4.1 Matrix Representation of Time-span Tree

In Fig 4(a), the first row of the matrix is the encoded pitch and duration and the second row is the connected parent branch number. The root branch has no parent branch to

which to connect, so the parent branch number is set to 0. Both the 2nd and 4th branches are connected to the 1st branch, but the branches of the time-span tree do not cross [1], indicating that the 4th is connected to the 1st at a position closer to the root. Notes that are missing due to reduction have blank pitches and durations on the matrix.

The 3rd row of the matrix is the branch priority. Since the branch priority is obtained from the time-span tree and the note duration, it is redundant information, but it is differentiated in this paper for clearer explanation.

4.2 Generation of Stepwise Reduction Data

Stepwise reduction data is generated by performing stepwise reduction in the order from the lowest priority branch. Applying a step-wise reduction to Fig. 4(a) reduces the notes in the 3rd branch, which has the lowest priority, to the 4th branch. The input of the Transformer model is ”d8 e8 e8 f8.” The output is ”d8 e8 r f16.” because the third note is absorbed on the right side. Next, when stepwise reduction is applied to Fig. 4(b), the note on 2nd branch with the second lowest priority is reduced to the note on 1st branch, and the input and output of the Transformer model are ”d8 e8 f16 -¿d16 l f16.”. Stepwise reduction data is created by repeating stepwise reduction until there is only one note left.

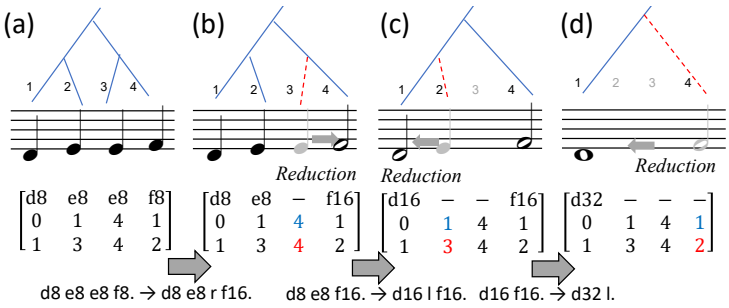

Fig. 4. Generation of stepwise reduction data

4.3 deepGTTM-IV: Reduction System

Figure 5 shows an overview of the reduction system. First, the input melody is converted into Matrix Representation of the time-span tree. In the initial state, no branches are connected, so the matrix has all 0 in the second row (Fig. 5(a)). Then the note sequence in the first row is sent to the Transformer model (Fig. 5(b)). The output of the Transformer model is reflected in the matrix in which the 3rd note is absorbed in the 4th note (Fig. 5(c)). Then (a) to (c) are iterated until there are no notes for reduction (Fig. 5(d)). Finally, a time-span tree is output (Fig. 5(e)).

The Transformer model may produce unexpected outputs from untrained inputs. In such a case, it may be difficult to proceed with the reduction process and our method

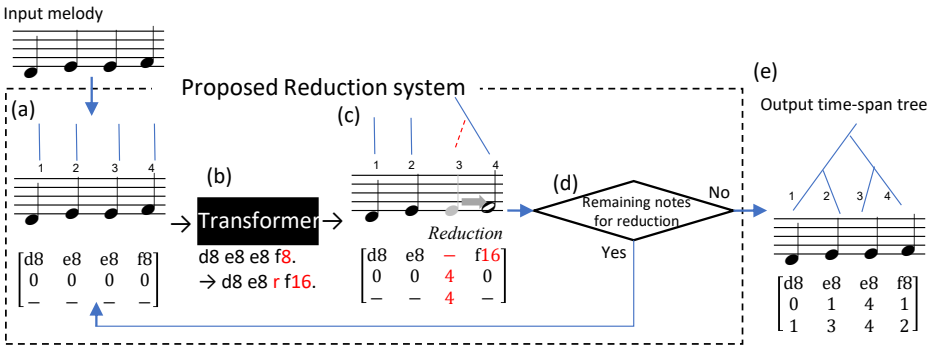

Fig. 5. Overview of reduction system

multiplies the initial duration of one note by a randomly chosen value between 2 and 16 and restarts the reduction process.

5 Experimental Results

We trained the Transformer model using 1,432,704 (= 7362 x 12 x 16) learning datasets created by data augmentation of 7362 stepwise reductions made from 270 pieces out of 300 pieces in the GTTM database. Accuracy was 0.99 when evaluated with 849 Stepwise reductions made from the remaining 30 pieces. Learning was carried out using Nvidia Quadro RTX5000 for laptops [17], and the learning time was seven hours.

We tried to acquire time-span trees for the remaining 30 pieces with deepGTTM-IV using the trained Transformer model, and we were able to acquire time-span trees for 29 pieces. The one remaining piece contained quintuplets and the output of the Transformer model was unexpected, so the notes could not be reduced.

6 Conclusion

Previous time-span analyzers could hardly obtain time-span trees without analysis errors, but we dramatically improved the analysis performance by learning step-wise reduction with the Transformer model. At the time of encoding the training data, by specifying which note to be reduced to the left or right will be absorbed, decoding becomes possible and a time-span tree can be obtained. As a result of experiments with 30 pieces, all time-span trees were obtained except one piece that contained quintuplets. We plan to conduct evaluation experiments with more pieces. In the case of quintuplets, septuplets, and higher multituplets, there is little data in the GTTM database and it is difficult to learn by the Transformer model, so we plan to increase the data of multituplets by data augmentation to improve performance.

Acknowledgements This work was supported by JSPS KAKENHI Grant number 21H03572.

References

1. Lerdahl, F., and Jackendoff, F.: *A generative theory of tonal music*. The MIT Press, Cambridge. MA (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.: ATTA: Automatic Time-span Tree Analyzer Based on Extended GTTM. In: *Proceedings of the 6th International Conference on Music Information Retrieval Conference* (ISMIR2005), pp. 358–365 (2005)
4. Hamanaka, M., Hirata, K., and Tojo. S.: FATTA: Full Automatic Time-span Tree Analyzer. In: *Proceedings of the 2007 International Computer Music Conference* (ICMC2007), Vol. 1, pp. 153–156 (2007)
5. 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 41st IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), pp. 276–280 (2016)
6. Groves, R.: Automatic Melodic Reduction Using a Supervised Probabilistic Context-Free Grammar. In: *Proceedings of the 17th International Society for Music Information Retrieval Conference* (ISMIR2016), pp. 775–781, New York (2016)
7. Hamanaka, M., Hirata, K., and Tojo. S.: Sigma GTTM 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)
8. Hamanaka, M., Hirata, K., and Tojo. S: Musical structural analysis database based on GTTM. In: *Proceedings of the 15th International Society for Music Information Retrieval Conference* (ISMIR2014), pp. 325–330 (2014)
9. Hamanaka, M., Isono, Y., Hirata, K., and Tojo. S: Web-based time-span tree editor and analysis database. In: *Proceedings of the 17th Sound and Music Computing Conference* (SMC2020), pp. 338–343 (2020)
10. Amari, S., Ozeki, T., Karakida, R., Yoshida, Y., and Okada., M.: Dynamics of Learning in MLP: Natural Gradient and Singularity Revisited. *Neural Computation*, 30(1), 1–33 (2018)
11. Marsden, A., Hirata, K., and Tojo. S: No Longer 'Somewhat Arbitrary': Calculating Salience in GTTM-Style Reduction. In: *Proceedings of the 5th International Conference on Digital Libraries for Musicology* (DLfM '18), pp. 26–33 (2018)
12. Hamanaka, M., Hirata, K., and Tojo. S.: Time-span Tree Leveled by Duration of Time-span. In: *Proceedings of the 15th International Symposium on Computer Music Multidisciplinary Research* (CMMR2021), pp. 155–164 (2021)
13. Pineda, F. J.: Generalization of Back-propagation to Recurrent Neural Networks. *Physical Review Letters*, 19(59), 2229–2232 (1987)
14. Hochreiter, S. and Schmidhuber, J.: Long Short-term Memory. *Neural Computation*, 9(8), 1735–1780 (1997)
15. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin. I: Attention Is All You Need. In: *Proceedings of the 31st Conference on Neural Information Processing Systems* (NIPS2017), pp. 6000–6010 (2017)
16. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, A., and Amodei, D., Language models are fewshot learners, arXiv preprint arXiv:2005.14165, 2020.
17. Nvida, "NVIDIA RTX in professional laptops.", Available: `https://www.nvidia.com/en-us/design-visualization/rtx-professional-laptops/`