CTCOMPOSER: A MUSIC COMPOSITION INTERFACE CONSIDERING INTRA-COMPOSER CONSISTENCY AND MUSICAL TYPICALITY

Hiromi Nakamura Tomoyasu Nakano Satoru Fukayama Masataka Goto
National Institute of Advanced Industrial Science and Technology (AIST)
{hiromi.nakamura, t.nakano, s.fukayama, m.goto}@aist.go.jp

ABSTRACT

This paper proposes a music composition interface that visualizes *intra-composer consistency* and *musical typicality* in order to support composers' decisions about their new pieces' style while keeping them aware of the self-expression balance between keeping their own composition style and breaking out of it. While there are many composition support systems that focus on generating or suggesting specific musical elements (e.g., melody or chord), our proposed interface feeds back the user's composition consistency and typicality to help users understand their own balance and how other composers tended to maintain some self-expression balances. To estimate the consistency and the typicality, we focused on monophonic melody (i.e., note sequence) as a musical element and modeled it by using a Bayesian topic n-gram model called the hierarchical Pitman-Yor topic model (HPYTM). By using proposed interface, named *CTcomposer*, the user can get comprehensive views of previous pieces by checking scatter plots of their consistency and typicality values. This interface also continuously updates and visualizes the consistency and typicality along the user input musical notes so that a piece the user is composing can be compared with previous pieces. The user can also raise or lower the consistency and typicality values as desired.

1. INTRODUCTION

Since the composition of music requires specialized skills, there are many studies on composition support and automatic composition. Such research has focused on the automatic generation and suggestion of musical elements that can be used to create a musical piece based on the composer's style. Indeed, researchers have mentioned that *"composers invest in each of their works a combination of new and inherited materials"* [1] and *"many composers create music precisely by imitating a given style to which they add their own constraints"* [2]. When composers starting to work on a new piece, they may often have two workflows. First, they consider how they position their new piece in relation to their past pieces and pieces by other composers. Second, they consider how to choose each musical note to meet with the concept and policy of their new piece.

	Intra-composer consistency	Musical typicality
High	Pros: Leads to the enhancement of personality Cons: Seems to result in similar pieces	Pros: Pieces have high musical validity Cons: Pieces lie buried among numerous others
Low	Pros: Leads to a new challenge for composers Cons: Difficult to utilize a composer's musical experience	Pros: Pieces seem novel to both composer and listener Cons: Pieces tend to become atypical of a musical genre

Table 1. Properties of consistency and typicality.

	Typicality: High	Typicality: Low
Consistency: High	Expression/style acceptable to the publis and the composer	Strong personality
Consistency: Low	New expression/style for the composer	Novel expression/style

Table 2. Properties of musical pieces based on the interaction between consistency and typicality.

The goal of our research is to support these workflows by visualizing the position of a new piece and suggesting musical notes. We achieve this goal by leveraging music information retrieval (MIR) techniques that make it possible to analyze composers' past pieces. The first contribution of this work is to propose the concepts of *intra-composer consistency* and *musical typicality* in order to support composers' decision in the workflows. The second contribution is to propose a composition support interface, named *CTcomposer,* that visualizes the above consistency and typicality and suggests musical notes that enable composers to control the visualized consistency and typicality values as desired (Figure 1).

We define the intra-composer consistency as the consistency among musical pieces by the same composer. It becomes high if a composer composes a piece that is similar to the composer's past pieces. The consistency value can be calculated by comparing the piece being composed with a set of the composer's past pieces. This comparison calculates how much the piece and the set of the pieces share common elements (e.g., keys, phrases, and rhythm patterns).

On the other hand, we define the musical typicality as the similarity between a musical piece and a set of musical pieces by all composers in a collection. It becomes high if a composer composes a piece that is similar to many other composers' pieces. A high typicality value indicates that the piece being composed tends to include expressions that are established in the collection.

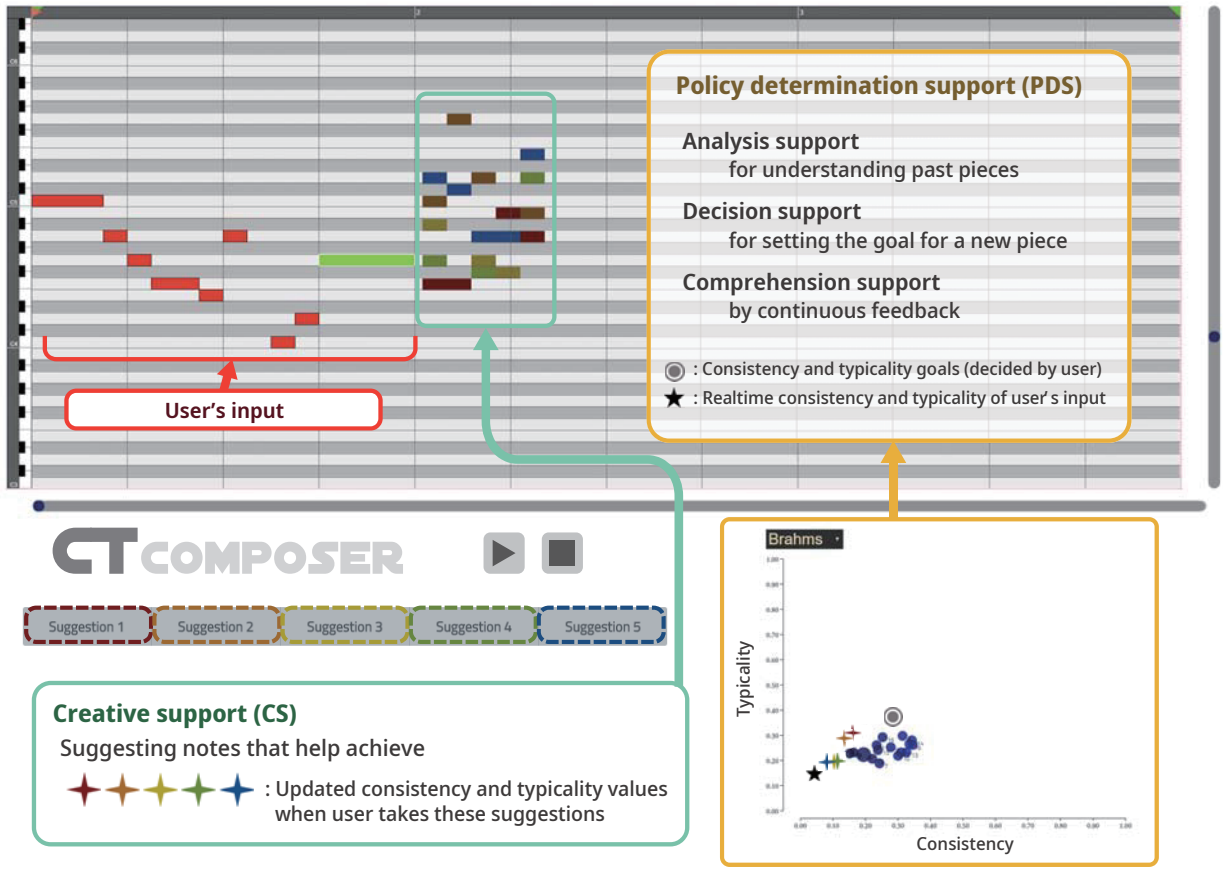

Figure 1. Overview of CTcomposer.

Table 1 summarizes pros and cons of this consistency and typicality. The disadvantages of each of them can be compensated by the advantages of other. If composers want to compose a low-consistency piece that they have not yet composed in a similar style, for example, they have to use musical expressions in a style new to them. This, however, was not easy because they could not use their past experiences. To support such a situation, high-typicality (i.e., musically acceptable) expressions can be suggested. The above compensation can be achieved by the interaction between the consistency and typicality as shown in Table 2. This interaction helps composers decide the position of a new piece since they can utilize the visualized consistency and typicality as if they were thinking 'Since I want to try a new style, I will try to make the consistency value lower than that of the previous piece.' It is, however, not easy for composers to be aware of the consistency and typicality in composing music because they cannot listen to or memorize all the musical pieces composed by them or other composers.

The CTcomposer interface therefore enables composers to leverage the interaction between the consistency and typicality, with a focus on the composition of monophonic melody sequences of classical vocal music. A user of CTcomposer can easily control the consistency and typicality of their composition by choosing from the suggested note candidates having various consistency/typicality values while looking at their visualized values. The consistency and typicality values are calculated by an *n*-gram based probabilistic generative model of the monophonic melody sequences by using state-of-the-art typicality estimation methods [3,4]. Since those methods focused on five musical elements (vocal timbre, musical timbre, rhythm, chord progression, and lyrics), to the best of our knowledge, this is the first work that applies the methods to the melody and shows their usefulness in supporting composition.

2. RELATED WORK

Many researchers have proposed methods/interfaces for supporting composers' creative practice. In this section, we introduce work on research related to composition interfaces that consider the style of the composer, automatic generation and support systems, and visual interfaces that provide comprehensive information about the content-creation process.

2.1 Automatic generation system based on a composer's style

FlowComposer [2] and the system proposed by Cope [5] consider the style of composers' past pieces and pieces by other composers. The idea that *'composers invest in each of their works a combination of new and inherent materials'* [1] is also similar to the idea underlying our proposal. CTcomposer has similarities to these systems, but they generate only high-consistency elements automatically. CTcomposer allows users to decide whether or not they set high consistency as a goal and suggests both high-consistency and low-consistency musical notes.

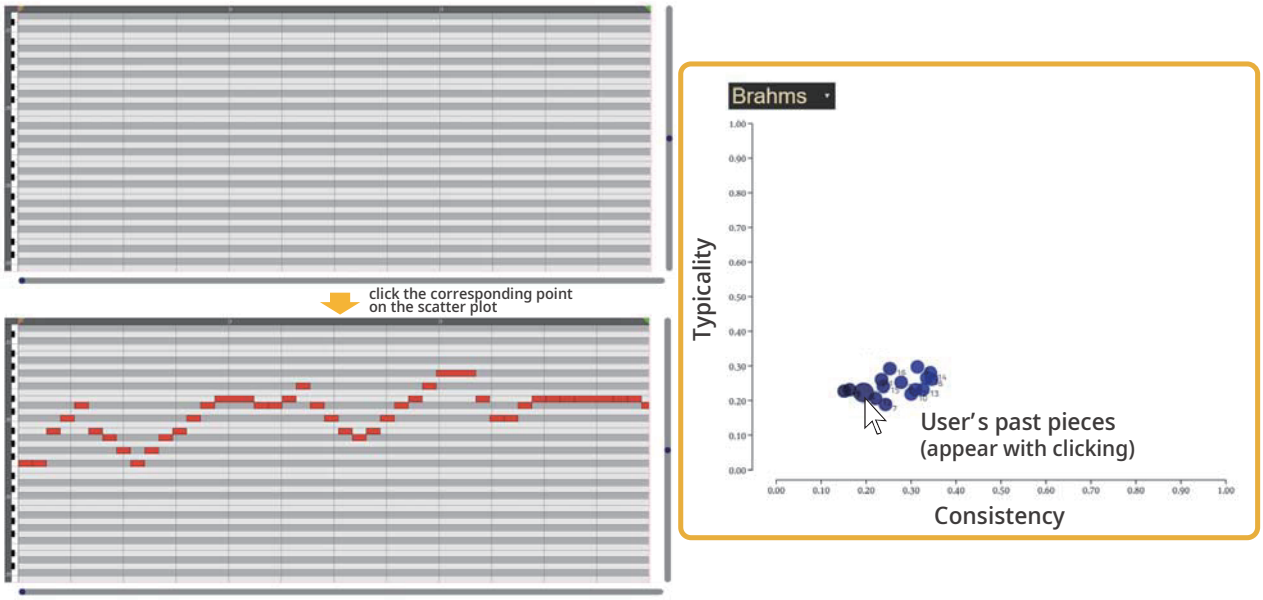

Figure 2. Melody preview of past pieces.

2.2 Interactive composition support system based on automatic generation and analysis

Automatic systems for composition utilized some models like probabilistic generative models, not only creating new pieces automatically but also helping the user's composition. OrpheusBB is an interactive composition system for creating melodies and arrangements while considering the musical consistency of the melody and the chord progression. [6]. This system generates chords and melodies based on lyrics and prosody. Users can edit the melodies and chords the system generates if they find something they do not prefer.

Shirai et al. proposed methods for generating melodies by using a variable-order Pitman-Yor language model (VPYLM) and using lyrics and chords progression as restrictions [7]. Spiliopoulou et al. utilized a topic model for generating both rich temporal structure and the complex statistical dependency [8]. Kitahara et al. utilized a hidden Markov models (HMM) for editing loop music, and their interface allows users to input climax tendency and edit musical contracture [9]. Lunanote also suggests the next musical notes and chords. It is a system focused on suggesting musical notes and chords in real time [10].

CTcomposer can be regarded as an interactive composition support system, and we also utilize a topic *n*-gram model for generating melodies. But those systems have tended to support particular points of a current piece, such as musical notes, chords, and phrases. In contrast, interface of CTcomposer is focused on the composer consistency of not only the current piece but also the user's past pieces.

2.3 Comprehensive interfaces for helping a user's creative practice

In the area of Human-Computer Interaction, several researchers have proposed systems that provide information in a way that helps give the user a broad perspective (like a plot does) and that helps specific workflows, like writing and editing. SidePoint presents information and pictures related to the presentation at the side of the main slide so that users can prepare presentation slides effectively [11]. Miyashita et al. examined what kinds of suggestions are effective during visual design and when these suggestions are effective [12]. Representational talkback is a concept that helps people get from thinking about what they should write to actually writing something [13].

Shneiderman [14] classified creative activities into the following four phases:

(1) Collect: learn from previous works stored in libraries, the Web, etc.
(2) Relate: consult with peers and mentors at early, middle, and late stages
(3) Create: explore, compose, and evaluate possible solutions
(4) Donate: disseminate the results and contribute to the libraries

The interface we propose can help in the Collect and Relate phases by calculating consistency and typicality with regard to past pieces and can help in the Create phase by suggesting musical notes.

3. CTCOMPOSER

The interface we propose here provides support in analyzing past pieces and suggests melodies. The analysis and suggestions are based on consistency and typicality.

3.1 Interface

CTcomposer has two main functions: policy determination support (PDS), and creative suggestion (CS). PDS provides three kinds of support: analysis support for understanding past piece, decision support for setting goals, and comprehension support giving continuous feedback on consistency and typicality.

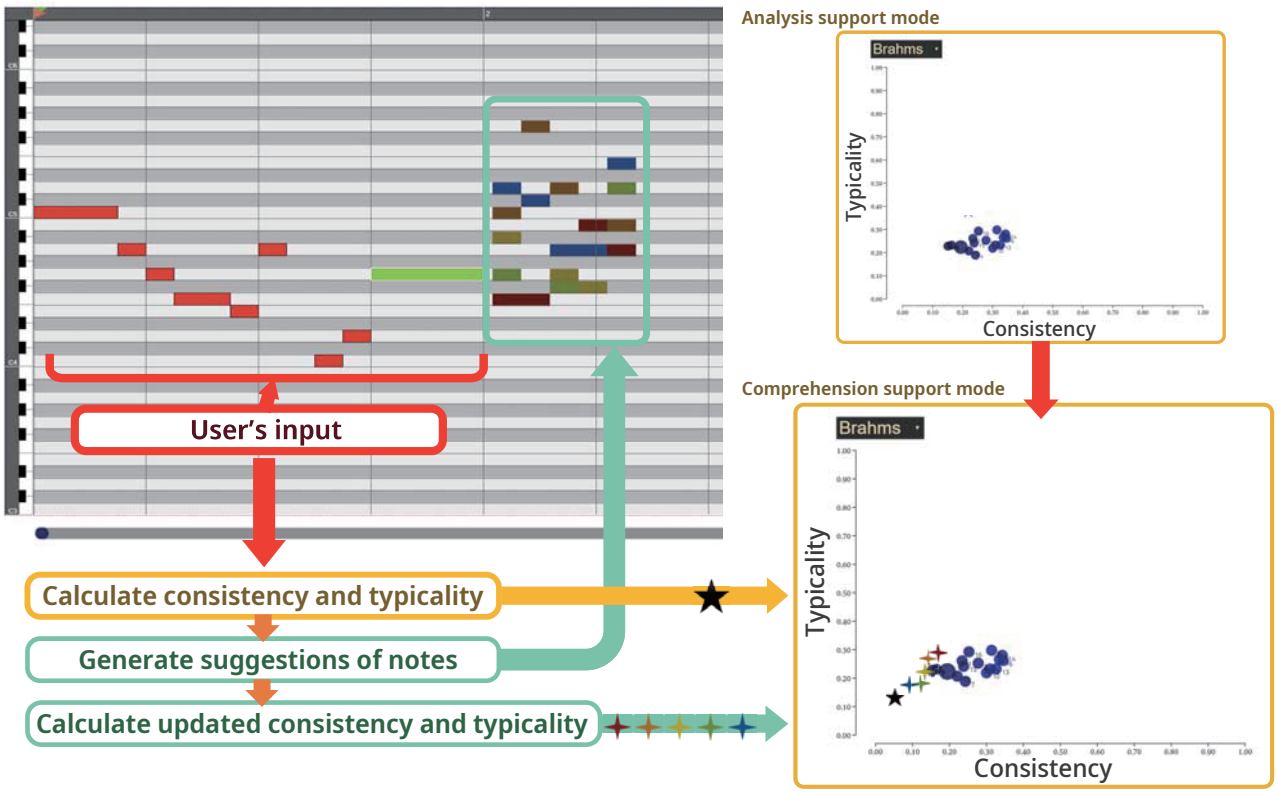

Figure 3. System flow with user's input.

CS also provides melody suggestion based on a probabilistic generative model and visualizes the consistency and typicality values based on the melody. These values are produced by combing the user's input and generated melodies and are updated continuously.

The user interface of CTcomposer thus consists of a visualization unit and note input unit. The visualization unit includes a control panel and scatter plot for checking consistency and typicality values. We designed the note input unit in the 'piano roll' style because CTcomposer supports melody composition. This unit is also used as a feedback area for the generated melody.

3.2 Scenario

Before the users start composing, they can input past pieces into CTcomposer and check the consistency and typicality scatter plot displayed in the lower right area of the interface (see Fig. 1, yellow rectangle). The X-axis represents the consistency value of each song, and the Y-axis represents the typicality value. Users can check the melody of a song by clicking the corresponding point on the scatter plot. The selected melody note is displayed on the piano roll (Figure 2).

Users can check the distribution of consistency and typicality of their past pieces and can consider the consistency and typicality goals of the piece they will compose. If their pieces tend to have high consistency, for example, they can try to compose pieces less consistent than their past pieces. The interface allows the user to set a goal by clicking on the scatter plot, and it visualizes as a gray double circle icon.

This current consistency and typicality are calculated from the first note of the inputted piece to the last. Model of the user's input is compared with a consistency model, which is all of the user's past pieces, and a typicality model consisting of many composers' past pieces. By checking the position and movement of the black star icon, users can compose while confirming that their input affects their pieces in the ways that they expect.

In addition, CTcomposer has a function visualizing the musical notes predicted from a probabilistic generative model because sometimes it may be hard to create phrases that can approach the target even if the user can perceive the gap between the target value and the current value. This function works automatically when a user inputs several musical notes into the piano roll user interface. Following the user's input, five melodies are generated and overlaid onto the piano roll with different colors. CTcomposer also calculates updated consistency and typicality, and these values are displayed on the scatter plot as colored crosses (Figure 3). The user can decide which suggestions they will choose by referring to both melody favorability and the suggestions' abilities to fill the gap. Of course, the user can input the next note without following suggestions. In this case, the current typicality and consistency are updated according to the user's input.

3.3 Calculation method

CTcomposer suggests generated melodies based on the user's melody by using a Bayesian topic n-gram model (HPYTM) [15]. In this section, we introduce methods of computing the consistency and typicality and generating suggestions of musical notes by using probabilistic models and topic distribution.

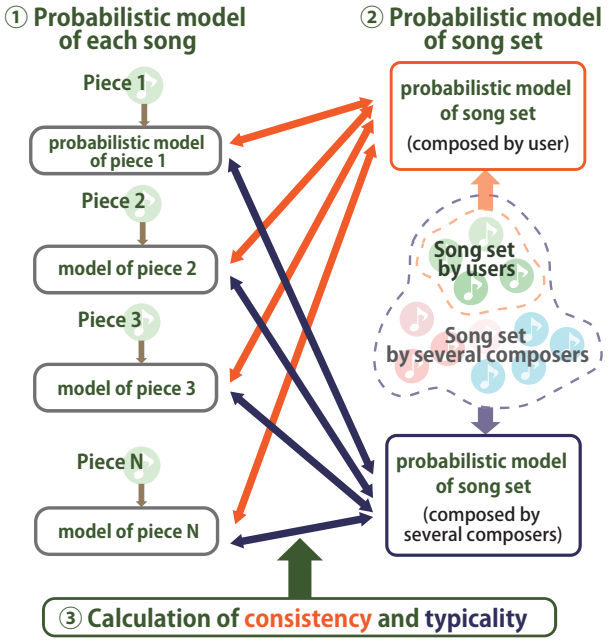

Figure 4. Method of calculating consistency and typicality based on different datasets.

3.3.1 Computing the consistency and typicality

We calculate consistency and typicality by comparing models of one song (including the user's work in progress) with a model of song set. As in our previous method [3,4], we compute the consistency and typicality of a song as compared to a set of other songs. We use the user's song set for calculating consistency and use a set containing the songs of many composers for calculating typicality.

Here we use HPYTM to calculate a topic distribution of each song [15] and train a probabilistic model that can compute the consistency and the typicality [3]. After we get probabilistic models for the song and song sets, we calculate the distance from the song's model and the song set's model [3] (Figure 4). We use as a melodic feature a sequence of MIDI note numbers. Since the model is trained without using note durations or onset intervals, such information is not included.

3.3.2 The suggestion of musical notes by HPYTM

As described earlier in this paper, CTcomposer suggests musical notes for supporting a user's composition. These notes are generated by a probabilistic model considering topic distribution we calculated using song set. We generate these musical notes on the basis of the conditional probability distribution $p(x|h)$ (x: note number, h: context, $h = x_{t-\infty} \ldots x_{t-2}, x_{t-1}$) because this distribution can consider previous notes through the generative process (Figure 5).

CTcomposer generates five suggestions for getting variations of updated consistency and typicality to reach the user's goal consistency/typicality. Usually, generated melodies tend to include more high-probability candidates than low-probability candidates. This influences the consistency value of the song being composed. Hence, we set 6 patterns of suggestions of musical notes by using a generative probability model from which the top candidates were eliminated one by one.

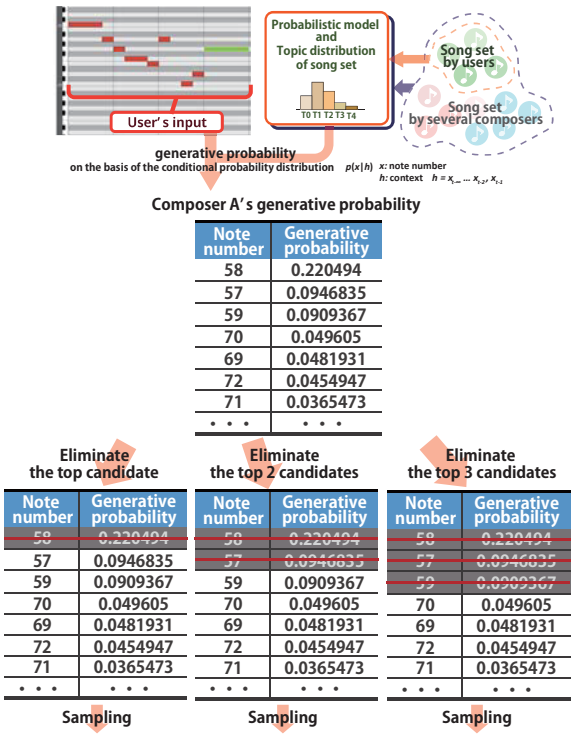

Figure 5. Generating melodies based on the dataset. Our system varies updated consistency and typicality by changing the number of high-probability candidates eliminated.

This helps users decides which suggestions to select because they can consider how they design whole songs and phrases: If they want to compose high-consistency piece, they select suggested melodies that generated all of the rankings, and they also choose melodies generated by low-probability sets in some parts for increasing and decreasing the consistency and typicality values in specific parts of the pieces they compose.

4. EXPERIMENT

CTcomposer provides suggestions based on probabilistic models for supporting the user's composition. We generated some kinds of suggestions by eliminating high-probability candidates (Figure 5). In this section, we examine whether we provide various updated consistency values by generating musical notes based on our method.

In this experiment, which was done as a simulation in which the users submitted their past pieces, we inputted as a song set existing songs of classical composers. All of the datasets were vocal songs in the classical genre, and in the experiment, we used 53 songs that were composed by Brahms (17 songs, 1851 total musical notes), Mozart (16 songs, 3873 total musical notes), and Schubert (21 songs, 5118 total musical notes). We extracted melody musical notes from these songs' standard MIDI files. We calculated consistencies in a supposed case in which the user accepted each generated musical note and plotted as transitions of consistencies values. An experimental approach using this kind of user behavior simulation has been used in previous studies [17,18].

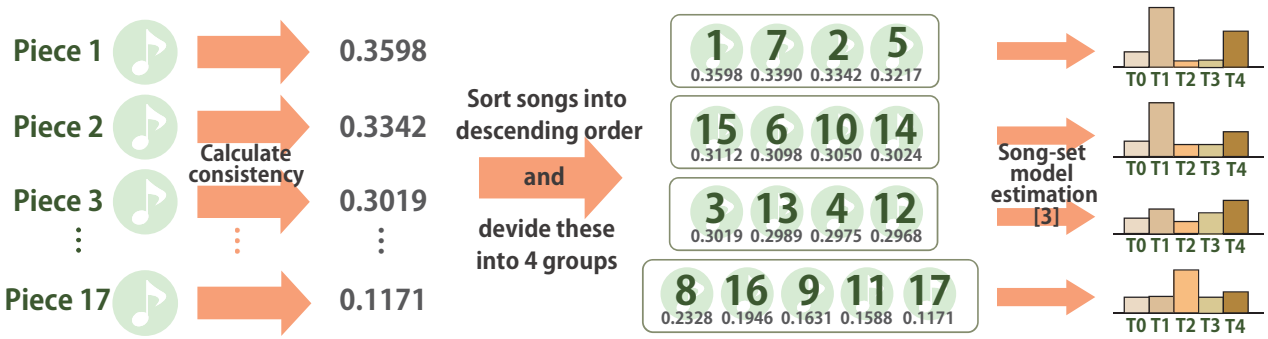

Figure 6. Experimental procedure of 4.1.2.

4.1 Datasets and experimental procedure

We used 17 Brahms songs as a dataset and trained a generative probabilistic model, generated suggestions of musical notes and calculated updated consistency by using the HPYTM. The number of topics was set to 5, and the model parameters of the HPYTM were trained by using the token-based Gibbs sampler [15] with 1000 iterations. We generated 1000 musical notes and calculated consistency each time 10 musical notes were added.

4.1.1 Experiment 1: Comparing updated consistency with the changing number of candidates eliminated from ranking

As described earlier in this paper, CTcomposer proposes several suggestions which have different updated consistency values by eliminating high-probability candidates (Figure 5). In the suggestions of musical notes based on the conditional probability distribution, we set 6 patterns of suggestions of musical notes by using a generative probability model from which the top candidates were eliminated one by one.

4.1.2 Experiment 2: Comparing updated consistency with the changing topic distribution

We also presumed that consistency of generated musical notes will become different when we use topic distributions of specific songs; for example, the topic distribution of groups containing high-consistency songs generates more high-consistency suggestions of musical notes than does the topic distribution of groups containing low-consistency songs. Hence, we generated 6 patterns of suggestions of musical notes by each 4 different topic distributions. We got 4 different topic distributions by dividing Brahms songs into 4 groups, in accordance with the ranking of consistency value of these songs (Figure 6).

4.1.3 Experiment 3: Calculating updated consistency by using suggestions of musical notes generated from another composer's probabilistic models and topic distribution

As following our hypothesis (if our method works well), probabilistic models and topic distribution may differ between composers, so it will not work well for increasing/decreasing updated values. Hence, we calculated the updated consistency value which is calculated by using musical notes generated with another composer's models and distribution for justify our method. We calculated this transition using 17 Brahms songs to calculate consistency and using 16 Mozart songs as the dataset for generating the suggested musical notes.

4.2 Results

Figure 7 and 8 are line graphs representing transitions of consistency values when the user continues accepting the system's suggestion of musical notes. The horizontal axis represents the number of musical notes added, and vertical axis represents the consistency of the piece. Below we describe the results of three experiments.

4.2.1 Result 1: Comparing updated consistency with the changing number of candidates eliminated from ranking

The top of Figure 7 shows the transition of consistency value when the simulated user keeps choosing our suggestion of musical notes. We found that we can get variations of consistency value after about 200 musical notes added, by adjusting elimination strategy (see the top of Fig 7, gray line). In addition, the consistency values of some melodies generated by eliminating the 1st to 6th highest probability candidates were lower than the value generated from the top 7 candidates.

4.2.2 Result 2. Comparing updated consistency with changing the topic distribution

The graph in the middle of Figure 7 shows the consistency values when we simulated a user who on accepting our suggestions method. The suggestions of musical notes were generated using the topic distribution which calculated 4 Brahms songs that had high consistency values. The result shows that the consistency values that were generated from the topic distribution of highest-consistency song set and used all sets of notes in the high-probability ranking had higher consistency values than those generated using topic distribution calculated by all songs and. On the other hand, as seen in the bottom of Figure 7, the consistency values generated using the topic distribution which calculated low-consistency song set had narrower ranges of values those generated using all the songs.

4.2.3 Result. 3 Calculating updated consistency by using the suggestion of musical notes generated from another composer's models and topic distribution

Figure 8 shows the transition of consistency value when the simulated user keeps choosing our suggested musical notes when the suggestions which were generated by using another composer's model and topic distribution.

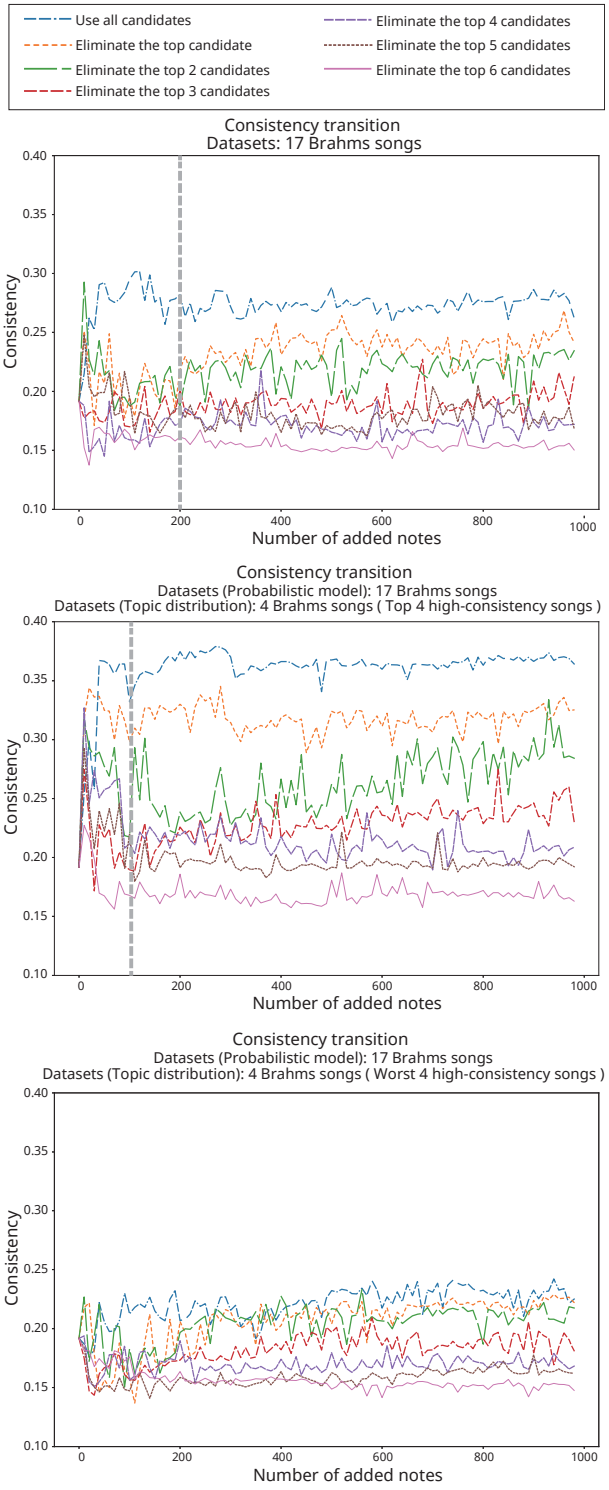

Figure 7. Consistency transition when the user accepted the conditional probability suggestions of musical notes. Top: results of experiment 1 using 16 Brahms songs. Middle: results of experiment 2 using topic distribution calculated by 4 high-consistency Brahms songs. Bottom: results of experiment 2 using topic distributions calculated by 4 low-consistency Brahms songs.

Each value of consistency increased in the first 10 notes. However, consistency values of each suggestion were not correlated with the number of eliminated candidates.

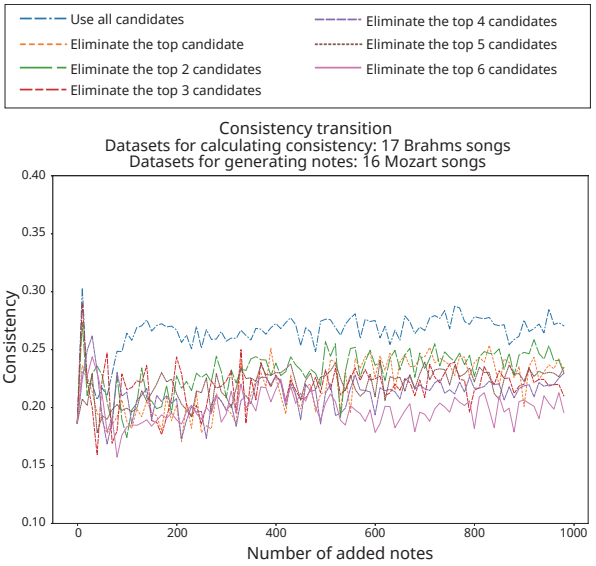

Figure 8. Results of experiment 3: Consistency transition calculated using suggestions of musical notes generated from another composer's

4.3 Discussion

As a result of our three experiments, we got three knowledge outcomes for the suggestions of musical notes. Firstly, we can provide several suggestions of musical notes which have various ranges of updated consistency by eliminating high-probability candidates when we generate notes by our model. However, the results show that we cannot get updated consistencies that are synchronized with the number of eliminated sets before adding about 200 musical notes. It seems this is impractical for melody composition because most songs have fewer than 200 musical notes.

Secondly, changing topic distribution for generating a suggestion of musical notes may solve this problem because most of the updated consistency values were synchronized with the number of eliminated sets before 100 musical notes were added. Furthermore, we found that musical notes suggested using the topic distribution calculated by 4 high-consistency songs can get richer ranges of consistency values. This shows that the suggestion system would work effectively when using topic distribution calculated by selecting several songs of a composer, especially high-consistency songs because the groups with low-consistency songs had narrower ranges of values than did the values of all songs.

And lastly, as we presumed, updated consistency for most of the values was not synchronized with eliminated sets if we added melodies generated using another composer's models and topic distribution. However, melodies that were generated using all of the sets of musical notes in the ranking by another composer have consistency values higher than those of other candidates. It cannot be immediately claimed that we can utilize this result for the suggestion, but this result suggests we can combine suggestions of musical notes by the users with suggestions of notes by others by keeping control of the consistency value.

5. CONCLUSION

In this paper, we proposed an interface, named CTcomposer, that supports users' composition with setting goals based on consistency and typicality. We also proposed methods for calculating consistency and typicality and proposed a suggestion method using a generative probability model made using datasets.

In future work we need to, as Papadopoulos mentioned [19], consider how to prevent unintended plagiarism in the suggestion system. Our method utilizes previous works of the user and thus does not cause plagiarism. We believe the suggestion based on typicality will become more common when using a melody phrase that is not specific to an existing work is not considered plagiarism. However, the current method cannot distinguish whether or not a phrase used often is actually typical of a composer. This is because method considers only the number of occurrences of musical notes. We will improve our system's ability to solve this problem by visualizing who used the suggestion of musical notes in past pieces and how many times that person used them.

This paper focused on monophonic melody, but this technique could be extended to other features and parts of the composition. In future work we will deal with visualization that shows different kinds of genres in one graph and a design experience in which the composer and system have a good relationship; composing is not left entirely up to the composer, and composers are encouraged. In addition, we will get listener feedback we can use for evaluating whether or not listeners also notice the composer's tendency. We will improve CTcomposer and its associated methods to encourage various composition activities.

Acknowledgments

This work was supported in part by JST ACCEL Grant Number JPMJAC1602, Japan.

6. REFERENCES

[1] D. Cope. Machine Models of Music, MIT Press, pp. 403–425, 1992.

[2] F. Pachet, P. Roy. Imitative leadsheet generation with user constraints. In Proc. ECAI 2014, pp. 1077–1078, 2014.

[3] T. Nakano, K. Yoshii, M. Goto. Musical typicality: how many similar songs exist? In Proc. ISMIR 2016, pp. 695–701, 2016.

[4] T. Nakano, D. Mochihashi, K. Yoshii, M. Goto. Musical similarity and commonness estimation based on probabilistic generative models of musical elements, International Journal of Semantic Computing, vol. 10, no. 1, pp. 27–52, 2016.

[5] D. Cope, An expert system for computer-assisted composition. Computer Music Journal, vol. 11, no. 4, pp. 30–46, 1987.

[6] T. Kitahara, S. Fukayama, S. Sagayama, H. Katayose, N. Nagata. An interactive music composition system based on autonomous maintenance of musical consistency. In Proc. SMC 2011, 2011.

[7] A. Shirai, T. Taniguchi. A Proposal of an Interactive Music Composition System using Gibbs Sampler. In Proc. (HCII'11), pp. 490–497, 2011.

[8] A. Spiliopoulou, A. Storkey. A topic model for melodic sequences, In Proc. ICML-12, pp. 1143–1150, 2012.

[9] T. Kitahara, K. Iijima , M . Okada , M. Yamashita. A loop sequencer that selects music loops based on the degree of excitement. In Proc. SMC 2015, pp. 435–438, 2015.

[10] J. Granger, M, Aviles, J, Kirby, A. Griffin, A. Yoon, J. R. Lara-Garduno, T. Hammond. Lumanote: a real-time interactive music composition assistant, IUI 2018, 2018.

[11] Y. Liu, D. Edge, K. Yatani. SidePoint: a peripheral knowledge panel for presentation slide authoring. In Proc. CHI 2013, pp. 681–684, 2013.

[12] H. Miyashita, K. Nishimoto, D. Miyata. How should a visual design process be supported?, AIDIA Journal, vol. 6, pp. 180–186, 2006.

[13] Y. Yamamoto, S. Takada, M. Gross, K. Nakakoji. Representational talkback: an approach to support writing as design. In Proc. APCHI 1998, pp. 125–131, 1998.

[14] B. Shneiderman. Creating creativity: user interfaces for supporting innovation, ACM Transactions on Computer-Human Interaction (TOCHI), vol. 7, no. 1, pp. 114–138, 2000.

[15] H. Noji, D. Mochihashi, D, Y. Miyao. Improvements to the Bayesian topic *n*-gram models. In Proc. EMNLP, pp. 1180–1190, 2013.

[16] K. Yoshii, M. Goto. A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis. In Proc. ISMIR 2011, pp. 645–650, 2011.

[17] E. Pampalk, T. Pohle, G. Widmer. Dynamic playlist generation based on skipping behavior, In Proc. ISMIR 2005, pp.634–637, 2005.

[18] J. P. V. Cardoso, L. F. Pontello, P. H. F. Holanda, B. Guilherme, O. Goussevskaia, A. P. C. da Silva. Mixtape: direction-based navigation in large media collections, In Proc. ISMIR 2016, pp. 454–460, 2016.

[19] A. Papadopoulos, P. Roy, F. Pachet. Avoiding Plagiarism in Markov Sequence Generation, In Proc. AAAI'14, 2014, pp. 2731-2737.