

T-Music: A Melody Composer based on Frequent Pattern Mining[†]

Cheng Long, Raymond Chi-Wing Wong, Raymond Ka Wai Sze
The Hong Kong University of Science and Technology
{clong, raywong, kwrsze}@cse.ust.hk

Abstract—There are a bulk of studies on proposing algorithms for composing the melody of a song automatically with algorithms, which is known as *algorithmic composition*. To the best of our knowledge, none of them took the lyric into consideration for melody composition. However, according to some recent studies, within a song, there usually exists a certain extent of correlation between its melody and its lyric. In this demonstration, we propose to utilize this type of correlation information for melody composition. Based on this idea, we design a new melody composition algorithm and develop a melody composer called T-Music which employs this composition algorithm.

Index Terms—frequent pattern mining, probabilistic automata, melody composition

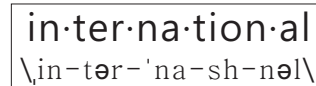
I. INTRODUCTION

Algorithmic composition [1] refers to the process of composing the melody of a song automatically by algorithms. Many approaches were proposed for algorithmic composition, Some representative approaches are *Markov models* [2], *generative grammars* [3], *transition networks* [4] and *Genetic algorithm* [5]. [1] gives a comprehensive survey on the approaches for algorithmic composition.

When a lyric is present in a song, algorithmic composition should consider not only the *temporal correlation* among all notes (or sounds) of the melody in the song but also the *lyric-note correlation* between the notes and the lyrics in the song. Most of the existing approaches only focused on the temporal correlation but no lyric-note correlation because they only studied algorithmic composition when a lyric is *absent* in a song.

The lyric-note correlation corresponds to the correlation between the *changing trend* of a sequence of consecutive notes and the *changing trend* of a sequence of consecutive (corresponding) words. The changing trend of a sequence of notes corresponds to a series of *pitch differences* between every two adjacent notes since each note has its pitch (or its frequency). The changing trend of a sequence of words corresponds to a series of *tone differences* between every two adjacent syllable since each syllable has its *tone*. For example, the English word “international” has 5 syllables. Besides, each syllable is spoken in one of the three kinds of *stresses* or *tones*, namely the *primary stress*, the *secondary stress* and the *non-stress*. The primary stress should sound with a higher

[†]The scope of this paper is related to frequent pattern mining, which matches two of the topics in the research track, namely “Data Mining and Knowledge Discovery: Algorithms” and “Data Mining and Knowledge Discovery: Applications”.



in·ter·na·tion·al
\,in-tər-'nā-sh-nəl\

Fig. 1. Tones of the word “International” (from merriam-webster dictionary)

frequency, the second one should sound with a lower frequency and the non-stress should sound with the least frequency. In Figure 1, we can see that the third syllable corresponds to the primary stress, the first syllable corresponds to the secondary stress and each of the other syllables corresponds to the non-stress. Tones appear in many languages in the world in addition to English. In Mandarin, there are four or five tones and each word has only one syllable. In Cantonese, there are six tones and each word also has only one syllable. Other languages with tones include Thai language and Vietnamese, etc.

Some recent studies [6, 7] found that the lyric-note correlation is very common in songs. [6] studied how to use this correlation for a foreigner to learn a new language. [7] studied the statistical information about this correlation. However, to the best of our knowledge, none of these existing studies use this correlation for algorithmic composition.

In this paper, we study the following problem. Given a lyric written in a language with different tones, we leverage the lyric-note correlation for melody composition and develop a melody composer called *T-Music*. There are two phases in T-Music. The first phase is a preprocessing phase which first finds lyric-note correlations based on a database containing a lot of existing songs each of which involves both its melody and its lyric by performing a *frequent pattern mining* task on this database. Then, based on the lyric-note correlations found, it builds a *Probabilistic Automaton* [8] (PA). The second phase is a melody composition phase which generates a melody given a lyric by executing the PA generated in the first phase.

T-Music has two obvious advantages over the existing algorithmic composition methods which do not consider the lyric-tone correlation. First, T-Music enjoys a richer knowledge source for melody composition. T-Music utilizes not only an existing song database as most existing methods did, but also utilizes the tone information of the given lyric. Second, T-Music is more user-friendly. Since a user who does not have much knowledge about music does not know how to choose the suitable melody composition algorithm. From a user’s perspective, it is more understandable to obtain a melody given the lyric written by him/her.

The contributions of this demonstration are as follows. First,

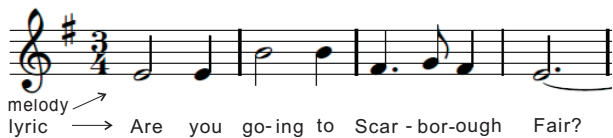


Fig. 2. A fragment of song (Scarborough Fair)

the idea of utilizing the lyric-note correlation for algorithmic composition is new. Second, based on this idea, we design a new algorithm for melody composition and develop a melody composer T-Music, which automatically generates a melody according to esthetic criteria.

In this demonstration, we first introduce our design of melody composition system employed by T-Music in Section II. Then, we present our melody composer T-Music in Section III. We conclude our demonstration with some future directions in Section IV.

II. DESIGN

In this section, we introduce the design of T-Music.

A. Background Knowledge

As we described before, each syllable is associated with a tone. Let T be the total number of tones. In this system, each tone is associated with a tone ID $\in [1, T]$. For example, in English, there are three possible tones where 1, 2 and 3 can be used to represent the tone IDs for the primary stress, the secondary stress and the non-stress, respectively. In Mandarin, there are 4 or 5 tones, and in Cantonese, there are 6 tones.

We briefly review some basic concepts in music theory with Figure 2. In this figure, the upper part corresponds to a piece of melody which is represented by a sequence of notes, and the lower part corresponds to a lyric.

A melody is represented by a sequence of notes. Each note is associated with its pitch, denoting the frequency of the sound for the note, and its duration, denoting how long the sound lasts for. We denote a note by a pair in the form of (pitch, duration).

A lyric is a sequence of words and each word is associated with a number of syllables. Note that each syllable is associated with a tone ID. We can construct a sequence of tone IDs for the lyric.

By combining the melody representation and the lyric representation, we denote a song in the form of a sequence of 2-tuples each in the form of (note, tone ID). We call such a sequence as an *s-sequence*. For a specific (note, tone ID)-pair p , we denote by $p.note$ its note element and by $p.tone$ its tone element.

B. Architecture

We present the architecture of T-Music in Figure 3.

In Phase I, we are given a song database, containing a lot of existing songs, and a tone look-up table, containing the mapping between the syllable of each word and the tone ID. For each song in the song database, T-Music performs the Tone Extraction utility on its lyric, and obtains the tone sequence and thus the *s-sequence*. Then, it mines all frequent patterns

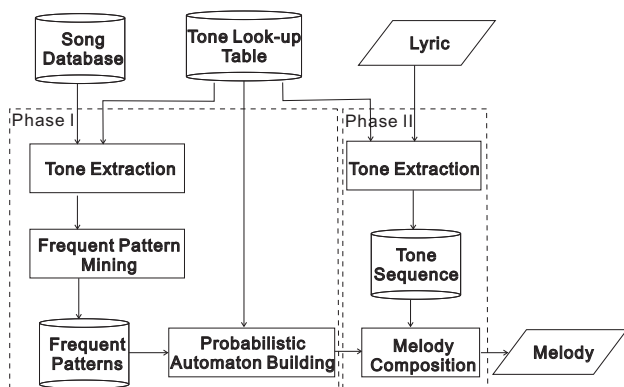


Fig. 3. Architecture of T-Music

based on all *s*-sequences found. Here, the frequent patterns corresponds to the lyric-note correlation. A detailed definition of these frequent patterns will be given later. Then, it builds a *Probabilistic Automaton (PA)* based on these frequent patterns.

In Phase II, we are also given the same tone look-up table and an input lyric written in a language. T-Music extracts the tone sequence from the input lyric via a Tone Extraction utility. Then, it performs the Melody Composition process by executing the constructed PA with the input of the extracted tone sequence, which outputs a melody.

In the following Section II-C, we will introduce three key utilities of T-Music, namely Frequent Pattern Mining, Probabilistic Automaton Building and Melody Composition, in detail.

C. Key Utilities

Frequent Pattern Mining. Let D be the set of *s*-sequences corresponding to the songs in the Song Database. Let S be a *s*-sequence. We define the *length* of S , denoted by $|S|$, to be the number of (note, tone ID)-pairs in S . We denote by $S[i, j]$ the *s*-sequence consisting of all (note, tone ID)-pairs which occur between the i^{th} position and the j^{th} position in S . For example, $S[1, m]$ corresponds to S itself, where m is the length of S . Given two *s*-sequences $S = ((n_1, t_1), \dots, (n_m, t_m))$ and $S' = ((n'_1, t'_1), \dots, (n'_{m'}, t'_{m'}))$, we define the *concatenation* between S and S' , denoted by $S \diamond S'$, as the *s*-sequence of $((n_1, t_1), \dots, (n_m, t_m), (n'_1, t'_1), \dots, (n'_{m'}, t'_{m'}))$. Besides, we say S' is a *sub-string* of S if there exists an integer i such that $S[i, i + m' - 1]$ is exactly S' , where m' is the length of S' .

We define the *support* of a *s*-sequence S wrt D to be the number of *s*-sequences in D that have S as its sub-string. Given a threshold δ , the Frequent Pattern Mining utility finds all *s*-sequences S with its support wrt D at least δ .

We adopt an existing algorithm [9] for finding frequent sub-sequence/substring mining for our purpose. For each frequent *s*-sequence S , we maintain its support, denoted by $S.\tau$.

Probabilistic Automaton Building. *Probabilistic Automaton* [8] (PA) is a generalization of *Non-deterministic Finite Automaton (NFA)* [10]. NFA is designed for lexical analysis in *automata theory*. Formally, NFA can be represented by a

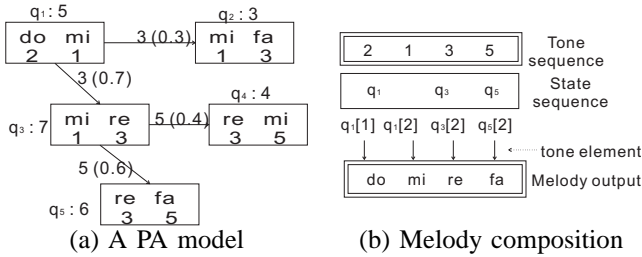


Fig. 4. A illustration example

5-tuple $(Q, \Sigma, \Delta, q_0, F)$, where (1) Q is a finite set of states, (2) Σ is a set of input symbols, (3) Δ is a transition relation $Q \times \Sigma \rightarrow P(Q)$, where $P(Q)$ denotes the power set of Q , (4) q_0 is the *initial* state and (5) $F \subseteq Q$ is the set of *final* (accepting) states.

PA generalizes NFA in a way such that the transitions in PA happen *with probabilities*. Besides, the initial state q_0 in NFA, which is deterministic, is replaced in PA with a probability vector v each of which entries corresponds to the probability that the initial state is equal to a state in Q . Thus, we represent a PA with a 5-tuple $(Q, \Sigma, \Delta, v, F)$, where Q, Σ and F have the same meanings as their counterparts in an NFA, and each transition in Δ is associated with a probability.

Let \mathcal{T} be the sequence of tone IDs extracted from the input lyric. An example of the sequence (called the tone sequence) can be (2, 1, 3, 5) (See the first row in Figure 4(b)).

In the following, we discuss the Probabilistic Automaton Building utility which constructs a PA represented by $(Q, \Sigma, \Delta, v, F)$.

(1) Q . We construct Q to be the set containing all s-sequences S that satisfy the following two conditions: (a) S has its length equal to l , where l is a user given parameter and (b) $\exists S' \in D$ such that S is a sub-string of S' .

(2) Σ . We construct Σ to be the set containing all tone IDs.

(3) Δ . We construct Δ as follows. At the beginning, we initialize Δ to be \emptyset . Then, for each pair of a state $q \in Q$ and a symbol $t \in \Sigma$, we perform the following two steps. First, we find a set of states, denoted by $Q_{q,t}$, such that each state q' in $Q_{q,t}$ satisfies the following: (1) $q'[1 : l - 1]$ is exactly the same as $q[2 : l]$ and (2) $q'[l].tone$ is exactly the same as t . Second, for each state $q' \in Q_{q,t}$, we create in Δ a transition from q to q' with the input of t and set its probability to be $q'.\tau / \sum_{q'' \in Q_{q,t}} q''.\tau$.

(4) v . For each state $q \in Q$, The probability that the initial state is q is set to be $q.\tau / \sum_{q \in Q} q.\tau$.

(5) F . We construct F as \emptyset . This is because the termination of the execution on the PA in our melody composition is not indicated by the final states. Instead, it terminates after all tone IDs in \mathcal{T} have been inputted, where \mathcal{T} is the sequence of tones extracted from the input lyric.

Figure 4(a) presents an instance of a PA. In the figure, we omit the duration for simplicity. There are 5 states, q_1, q_2, \dots, q_5 , each represented by a box. The number next to each state is the support of its corresponding s-sequence, e.g., $q_1.\tau = 5$. The arrow from a state to another means a transition and the number along the arrow is the input symbol

in Σ corresponding to the transition. Besides, the number within the parentheses is the probability associated with the corresponding transition.

Melody Composition. We generate the melody by executing the PA constructed by the Probabilistic Automaton Building utility with the input of the tone sequence extracted from the input lyric, i.e., \mathcal{T} . Specifically, let (q_1, q_2, \dots, q_n) be the sequence of resulting states when executing the PA with \mathcal{T} as the input. Then, the melody generated by T-Music, which is a sequence of notes, is represented by $(q_1[1].note, q_1[2].note, \dots, q_1[l].note) \diamond (q_2[l].note) \diamond (q_3[l].note) \dots \diamond (q_n[l].note)$. Note that $q_i[2 : l]$ is exactly the same as $q_{i+1}[1 : l - 1]$ since there exists a transition from q_i to q_{i+1} in Δ for $1 \leq i \leq n - 1$.

There is one remaining issue with this Melody Composition utility. Specifically, during the execution process on the PA, the following scenario might occur. There exist no transitions from the current state, says q , to other states with the current input tone ID, says t , i.e., $\Delta(q, t)$ is an \emptyset . Thus, in this case, the execution process cannot proceed. To fix this issue, in T-Music, we choose to select the state q' in Q such that (1) $q'[1 : l - 1]$ is the most similar to $q[2 : l]$, (2) $q'[l].tone$ is exactly the same as t and (3) $\Delta(q', t)$ is non-empty. The similarity measurement adopted in T-Music is the common *edit distance* measurement [11] between two strings.

We illustrate Melody Composition utility by executing the PA as shown in Figure 4(a) with the input of the tone sequence as shown in Figure 4(b). Suppose it chooses state q_1 as the initial state. After that, the current state is q_1 and the current input symbol is 3 (tone IDs 2 and 1 are involved in state q_1). At this moment, the next state could be either q_2 (with the probability equal to 0.3) or q_3 (with the probability equal to 0.7). Suppose it proceeds at state q_3 . Now, the current input symbol is 5. Further assume that it chooses q_5 as the next state. Since all tone IDs in the tone sequence have been inputted, the execution process stops. As a result, the sequence of resulting states is (q_1, q_3, q_5) and thus the melody generated is $(q_1[1].note, q_1[2].note, q_3[2].note, q_5[2].note)$, which is simply (do, mi, re, fa) with the duration information.

D. Miscellaneous Issues

There are some advanced concepts related to music theory. We also considered them for melody composition. Some of them are listed as follows. For the sake of space, we just highlight the major ideas and the details are skipped.

Harmony Rule. Two examples of harmony rules are the *chord progression* and the *cadence*. For the sake of space, we illustrate the rules with the cadence. Each song can be broken down into *phases*. We can regard a phase as a sentence in a language. In Music Theory, each phase ends with a *cadence*. A formal definition is skipped here. A cadence is a certain kind of patterns which describe the ending of a phase. It is just like a full-stop or a comma in English. According to the concept of cadence, the last few notes at the end of each phase must come from some particular notes. T-Music generates notes at the end of each phase according to this cadence principle. In

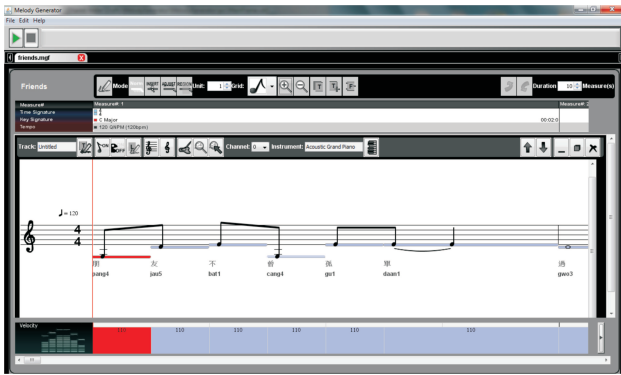


Fig. 5. Main interface of T-Music

particular, when we generate the notes at the end of a phase, we consider all the notes related to the cadence instead of all possible notes.

Rhythm. We consider the rhythm component for generating the melody. For example, the last note of a phase should be longer. The rhythm of a phase is similar to the rhythm of some of the other phases.

Coherence. In a song, one part in the melody is usually similar to the other part so that the song has a coherence effect. T-Music also incorporates this concept. Specifically, whenever we generate another phase for the melody, we check whether some portions of the melody generated previously can be used to generate the new portions of the melody to be composed automatically. If yes, we use some existing portions of the melody for the new portions. The criterion is to check whether each existing portion of the melody together with the portion of the lyric can be found in the frequent patterns mined in Phase 1.

Vocal Range There were some studies that the *vocal range* of a human is bounded (e.g., at most two octaves). The vocal range is the measure of the breadth of pitches that a human voice can sing. Based on the vocal range, T-Music restricts the all possible choices of notes to be generated whenever it executes the PA.

III. SOFTWARE

A. User Interface

Figure 5 represents the main interface of T-Music. To compose a melody, one needs the following three steps.

Lyric input & Parameter specification. This step is to input the lyric and specify some parameters. Specifically, one can click “File → Compose Melody” and then a dialog box pops up, in which the user can input the lyric and specify the parameters related to music, e.g., Key Signature, Time Signature and Instrument. Besides, this dialog provides the option for extracting the tone IDs from the input lyric.

Melody Composing & Editing. This step is for melody composition and melody editing. Based on the input lyric and parameters in the first step, T-Music automatically composes a melody. For example, based on the lyric and parameter information filled in the dialog box described above, T-Music

composes a melody as shown on the white board of the main interface in Figure 5. After the composition process, the user has the options for *playing* and *editing* the composed melody in an interactive way.

Melody Export. This step is to export the composed melody together with the lyric as a MIDI file. This could be done by choosing the “File → Export MIDI” command.

Besides, as will be demonstrated, T-Music also provides other functionalities such as frequent pattern browsing based on the song database, voice track management and different modes for playing and editing the generated melody.

B. Demonstrations

We demonstrate our T-Music by composing a nice melody for a prepared lyric. The demo can be found at <http://www.cse.ust.hk/~raywong/paper/MelodyGen.mpeg>.

IV. CONCLUSION

Motivated by the fact that few of the existing algorithmic composition methods took the lyric into consideration while there exists lyric-note correlation, we propose to utilize this correlation information for melody composition.

There are several interesting research directions related to our work in this demonstration. First, in our melody composition algorithm, to utilize the lyric-note correlation information, which is captured by frequent patterns, we choose to build a Probabilistic Automaton. However, we believe that Probabilistic Automaton is not the only option. In fact, one can consider exploring other models for this purpose. Another interesting direction is to seek for other concepts for capturing the above correlation information in addition to the concept of frequent patterns as we did in this demonstration.

Acknowledgements: We are grateful to the anonymous reviewers for their constructive comments on this paper. The research was supported by FSGRF12EG50.

REFERENCES

- [1] G. Nierhaus, *Algorithmic composition: paradigms of automated music generation*. Springer Verlag Wien, 2009.
- [2] F. P. Brooks, A. Hopkins, P. G. Neumann, and W. Wright, “An experiment in musical composition,” *Electronic Computers, IRE Transactions on*, no. 3, pp. 175–182, 1957.
- [3] F. Lerdahl, R. Jackendoff, and R. S. Jackendoff, *A generative theory of tonal music*. The MIT Press, 1996.
- [4] D. Cope, *Experiments in musical intelligence*. AR Editions Madison, WI, 1996, vol. 1.
- [5] A. Horner and D. E. Goldberg, “Genetic algorithms and computer-assisted music composition,” *Urbana*, vol. 51, no. 61801, p. 14, 1991.
- [6] S. Qin, S. Fukayama, T. Nishimoto, and S. Sagayama, “Lexical tones learning with automatic music composition system considering prosody of mandarin chinese,” in *Second Language Studies: Acquisition, Learning, Education and Technology*, 2010.
- [7] M. Schellenberg, “Singing in a tone language: Shona,” in *Proceedings of the 39th Annual Conference on African Linguistics*, 2009.
- [8] M. O. Rabin, “Probabilistic automata*,” *Information and control*, vol. 6, no. 3, pp. 230–245, 1963.
- [9] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, “Sequential pattern mining using a bitmap representation,” in *KDD*, 2002.
- [10] M. O. Rabin and D. Scott, “Finite automata and their decision problems,” *IBM journal of research and development*, vol. 3, no. 2, pp. 114–125, 1959.
- [11] M. J. Atallah, *Algorithms and theory of computation handbook*. CRC, 1999.