2015 IEEE 15th International Conference on Data Mining Workshops

Trend-MC: A Melody Composer by Constructing from Frequent Trend-based Patterns

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*—*Algorithmic composition* refers to the process of composing a melody automatically using algorithms. A bulk of methods have been proposed for this task. Among them, a novel idea is to utilize the correlation information between the pitches of melodies and the tones of lyrics for melody composition. Unfortunately, the existing method adopting this idea suffers from several severe shortcomings and thus the merits of the above idea are not fully utilized. In this paper, we propose a new technique to capture the above correlation information based on the concepts of *pitch trends* and *tone trends*. Based on this technique, we design a new algorithm called *Trend-MC* for melody composition which avoids the shortcomings. We also developed a software with the Trend-MC algorithm as its core. We demonstrate that the software could compose nice melodies with the input of lyrics.**

I. Introduction

Algorithmic composition which refers to the process of composing a melody automatically using algorithms has attracted much attention in the literature [1-4]. The main idea of these methods is to use the *temporal correlation* among all notes (sounds) of the songs in an existing song database for melody composition. They differ from one another by using different models for capturing the correlation information.

More recently, [5] proposed a new idea for melody composition when a lyric is present. Given an inputted lyric, they proposed to generate a melody for this lyric by utilizing the correlation information between the notes of existing melodies and the tones of the corresponding lyrics, which is shown to commonly exist [6]. This idea [5] is nice since it utilized an additional information source (i.e., lyrics) compared to previous melody composition methods.

Unfortunately, the method developed in [5] that adopted this idea has several shortcomings. The correlation information between the notes of the melodies and the tones of the lyrics is captured by *frequent patterns* called *frequent s-sequences*. Then, these frequent s-sequences are used to build a *Probabilistic Automaton* (PA) [7], and an execution of the PA with the lyric as input corresponds to the melody composition process. This method has the following shortcomings.

- Only those frequent patterns (i.e., s-sequences) that have a *fixed* length equal to a user-given parameter l are allowed to be used. This is not desirable since the knowledge embedded in all other frequent patterns of lengths not equal to l is wasted.
- During the melody composition process, the two frequent patterns that are used adjacently have to *overlap* with each other to a large degree ($l-1$ out of l) which poses a fairly strict condition. When this condition cannot be satisfied, the melody is composed more randomly which degrades the quality of the resulting melody considerably.
- The PA used in [5] usually involves a *tremendous* number of states, which incurs a fairly expensive computation cost. Specifically, in the setting that we have m songs each with n_i ($1 \leq i \leq m$) pitches, the number of states could be $\sum_{i=1}^{m}(n_i - l + 1)$ in the worst case.

In this paper, we adopt the same idea of utilizing the correlation information between the notes of melodies and the tones of lyrics as in [5] but use a new technique for capturing it. Instead of capturing the above correlation information with sequences of *absolute pitches* and *absolute tones* (utilized by [5]), we capture it with *pitch trend* (which contains a sequence of differences between adjacent pitches) and *tone trend* (which contains a sequence of differences between adjacent tones). This new technique gives a better interpretation of melodies, resulting in a better way of capturing the correlation information between melodies and lyrics. Since a melody, by its nature, is more like a sequence of changing pitch differences (e.g., higher, lower or stay) which corresponds to a *pitch trend* but not a sequence of *absolute pitches*. For example, for the *same* melody, we can play it with Piano in different ways (e.g., C-major or G-major), which correspond to the *same* sequence of pitch trends, but *different* sequences of absolute pitches.

Based on this new technique, we develop an algorithm called *Trend-MC* for melody composition, which avoids all the aforementioned shortcomings of [5] as illustrated as follows.

- All mined frequent patterns are allowed to be used in the melody composition process. That is, there is no restriction on the lengths of the frequent patterns and thus the correlation knowledge could be fully utilized.
- In the melody composition process, there is no requirement for the two frequent patterns that are used adjacently to be overlapped. We emphasize here that this feature does not come with the sacrifice of the smoothness of the composed melody since the frequent patterns used are based on pitch trends but not absolute pitches.
- It is not based on a PA model and thus it does not suffer from an expensive computation cost.

In the following, we introduce the Trend-MC algorithm in Section II and the software developed based on Trend-MC in Section III. We conclude the paper in Section IV.

978-1-4673-8493-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDMW.2015.165

II. The Trend-MC Algorithm

A. Preliminaries

A *song* usually consists of two parts, a *melody* and a *lyric*. A melody corresponds to a sequence of *notes* and each note has its *pitch* (i.e., how high the frequency of the sound is) and its *duration* (i.e., how long the sound lasts for). Note that each pitch corresponds to a frequency value (a real number). In this paper, we focus on the pitch part for illustration. Similar techniques can be applied to the duration part. Thus, each melody is associated with a sequence of pitches. In this paper, we are interested in the *pitch trend* information of a melody. Given a sequence of n pitches of a melody, $(p_1, p_2, ..., p_n)$, we capture its *pitch trend* with a sequence of $n-1$ "pitch differences", $(p_2 - p_1, p_3 - p_2, ..., p_n - p_{n-1})$.

In music theory, a common way of denoting pitches is to use so-fa names (e.g., do, re, mi, fa, so, la and ti). Given two so-fa names x and y, the pitch difference between x and y, denoted by $x - y$, is measured by the number of *semi-tones* between them (e.g., the pitch difference between do and re is 2 semi-tones which we denote by $re - do = +2$, and the pitch difference between re and do is still 2 semi-tones but we denote by $do - re = -2$). For example, given a sequence of pitches (mi, fa, so), we know that its corresponding pitch trend is $(+1, +2)$ (since $fa - mi = +1$ and $so - fa = +2$). Thus, in this paper, we assume that each pitch trend could be represented by a sequence of integers.

A lyric corresponds to a sequence of words, each of which has several *syllables* (phonological "building blocks"). For example, word "water" has two syllables: *wa* and *ter*. Each syllable has a corresponding *tone* which means the pitch in which the syllable is pronounced. Different languages have different sets of tones. For example, in English, there are three tones (i.e., primary stress, second stress, and non-stress) [5], and in Mandarin, there are five tones. Thus, each lyric is associated with a sequence of tones.

Similar to a melody which has a pitch trend, each lyric has its *tone trend*, which corresponds to a sequence of "tone differences" between adjacent tones in the sequence of the tones of the lyric. Note that we can calculate the *pitch* of each tone and capture the tone difference between two tones with their pitch difference. For simplicity, we round each tone difference to the nearest integral value. Thus, same as the pitch trend, we safely assume that each tone trend could be represented by a sequence of integers.

Each melody is composed of a number of *sentences* (or formally *phrases*) (like sentences in an article). In this paper, for illustration purpose, we regard the whole melody as a single sentence. The techniques to be described can also be easily extended to multiple sentences.

Next, we describe two kinds of pairs, namely a *(cadence, tone difference)-pair* and a *(pitch trend, tone trend)-pair*.

1) (cadence, tone difference)-pair: The (cadence, tone difference)-pair is used to describe the correlation between the melody part and the lyric part at the end of the whole song. In music theory, the ending two chords (or simply pitches) of a melody is called the *cadence* of the melody. In other words, each melody has a cadence. Given a cadence cad, we denote its first pitch by $cad[1]$ and its second pitch by $cad[2]$, i.e., $cad = (p_1, p_2)$ has $cad[1] = p_1$ and $cad[2] = p_2$.

It is a common knowledge in music theory that the cadence part of a melody should not be composed in a relatively free way as one does for the non-cadence part of the melody. Instead, there exists a *cadence pool* from which the cadence of a melody should come from. Motivated by this, in this paper, we propose to first collect the set of common cadences as the cadence pool and then use the "correlation information" between the cadences of existing melodies and the ending tone differences of corresponding lyrics for composing the cadence part of a new melody. Before we introduce how to capture this correlation information, we give the definition of "(cadence, tone difference)-pair" first.

Definition 1 ((cadence, tone difference)-pair): A (cadence, tone difference)-pair is a pair in the form of $(cad, \triangle_t)$ where cad is a cadence and $\triangle_t$ is a tone difference. □

In the following, for simplicity, we use "cadence-pair" to refer to "(cadence, tone difference)-pair".

Given a cadence-pair cp, we denote the cadence in cp by $cp.cad$ and the tone difference in cp by $cp.toneDiff$.

Let $\mathcal{D}$ be the song database and s be a song in $\mathcal{D}$. We say that s *owns* the cadence-pair $(cad, \triangle_t)$ if cad is the cadence of s and $\triangle_t = t_n - t_{n-1}$ where t_n is the last tone of the lyric of s and t_{n-1} is the second last tone of the lyric of s. Note that a cadence could be owned by multiple songs.

Given a cadence-pair cp, we define its *support*, denoted by $sup(cp)$, to be the number of songs in $\mathcal{D}$ that own cp.

Now, we define the concept called "frequent cadence-pair" which captures the correlation information between the cadences of melodies and the ending tone differences of lyrics.

Definition 2 (Frequent Cadence-Pair): A cadence-pair cp is said to be a *frequent cadence-pair* if $sup(cp) \geq \tau_{cp}$ where τ_{cp} is an integer and is a user-given parameter. □

As will be illustrated in Section II-B, we mine all frequent cadence-pairs for composing the cadence part of a melody.

2) (pitch trend, tone trend)-pair: The (pitch trend, tone trend)-pair is used to describe the correlation between the melody part and the lyric part at any non-end part of the whole song. Each song has its melody associated with a pitch trend and its lyric associated with a tone trend. In this paper, we mine the "correlation information" between these two trends, and then utilize it for composing the non-cadence part of a melody. Before we introduce how to capture this correlation information, we give the definition of "(pitch trend, tone trend)-pair" first.

Definition 3 ((pitch trend, tone trend)-pair): A *(pitch trend, tone trend)-pair* is a pair (P, T), where P is a pitch trend and T is a tone trend with the same size as P. □

In the following, for simplicity, we use "trend-pair" to refer to "(pitch trend, tone trend)-pair".

Let $tp = (P, T)$ be a trend-pair. We say that the *size* of tp, denoted by $size(tp)$, is $|P|$ $(= |T|)$. Besides, we denote the pitch trend of tp by $tp.P$ and the tone trend of tp by $tp.T$.

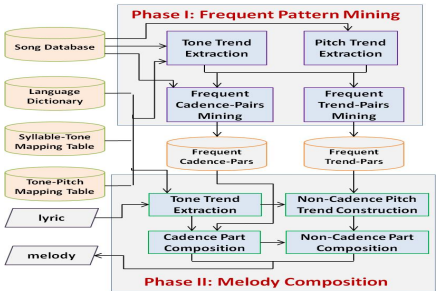

Fig. 1. The architecture of Trend-MC

We denote by $tp[i:j]$ $(1 \leq i < j \leq size(sp))$ the trend-pair of $(P[i:j], T[i:j])$, where $P[i:j]$ $(T[i:j])$ represents the sub-sequence of P (T) that contains the elements between the i^{th} position and the j^{th} position.

Given two trend-pairs $sp_1 = (P_1, T_1)$ and $sp_2 = (P_2, T_2)$, we say that sp_1 *contains* sp_2, denoted by $sp_1 \ni sp_2$, if there exists two integers i and j $(1 \leq i < j \leq size(sp_1))$ such that $sp_1[i:j]$ is exactly sp_2, i.e., $P_1[i:j] = P_2$ and $T_1[i:j] = T_2$.

Given $s \in \mathcal{D}$, we denote by $etp(s)$ the *embedded trend-pair* of s in the form of (P, T) where P is the pitch trend of the melody of s and T is the tone trend of the lyric of s.

Let $tp = (P, T)$ be a trend-pair. We define the *support* of tp denoted by $sup(tp)$, to be the number of songs in $\mathcal{D}$ each of which has its embedded trend-pair containing tp, i.e., $sup(tp) = |\{s \in \mathcal{D} | etp(s) \ni tp\}|$.

Now, we are ready to define the concept of "frequent trend-pair" which captures the correlation information between the pitch trends of melodies and the tone trends of lyrics.

Definition 4 (frequent trend-pair): A trend-pair tp is said to be a *frequent trend-pair* if $sup(tp) \geq \tau_{tp}$ where τ_{tp} is an integer and is a user-given parameter. □

As will be illustrated in Section II-B, we mine all frequent trend-pairs for composing the non-cadence part of a melody.

B. The Trend-MC Algorithm

Trend-MC involves two phases. The first phase is called *Frequent Pattern Mining* which mines two types of frequent patterns from the song database. Roughly speaking, these frequent patterns correspond to the correlation information between pitch trends (from existing melodies) and tone trends (from existing lyrics). The second phase is called *Melody Composition* which takes a (new) lyric as input and composes a melody for this lyric by using the frequent patterns mined in the first phase. Figure 1 shows the architecture of Trend-MC, which will be described in detail next.

1) Phase I: Frequent Pattern Mining: In this phase, we mine two types of frequent patterns: frequent cadence-pairs and frequent trend-pairs.

Before the mining procedures, we first collect a set $\mathcal{T}$ of trend-pairs each of which is embedded by a song in $\mathcal{D}$ as follows. We initialize $\mathcal{T}$ to be $\emptyset$ at the beginning. Then, for each song $s \in \mathcal{D}$, we extract the pitch trend of the melody of s as P and extract the tone trend of the lyric of s as T, and then add a trend-pair (P, T) into $\mathcal{T}$. The task of extracting the pitch trend of a melody is easy and all we need do is to compute the pitch differences between adjacent pitches in the sequence of the pitches of the melody. The task of extracting the tone trend of a lyric could be done with three steps. First, we construct a sequence of syllables corresponding to the sequence of words in the lyric by utilizing a "language dictionary". Second, we construct a sequence of tones corresponding to the constructed sequence of syllables by utilizing a "syllable-tone mapping table" which maps each syllable to a tone and could be collected from the Web. Third, we construct a sequence of tone differences by computing the pitch differences between adjacent tones in the constructed sequence of tones by utilizing a "tone-pitch mapping table" which maps each tone to a pitch value and could also be collected from the Web.

We then mine two types of frequent patterns using $\mathcal{T}$.

Frequent Cadence-Pairs Mining. Let $\mathcal{C}$ denote the set of frequent cadence-pairs to be mined, which is initialized as $\emptyset$ at the beginning. For each song $s \in \mathcal{D}$, we do two steps. First, we retrieve the cadence-pair cp that is owned by s (this could be easily done with the knowledge of $\mathcal{T}$). Second, we have two cases. Case 1: $cp \notin \mathcal{C}$. We add cp into $\mathcal{C}$ and set $sup(cp)$ to be 1. Case 2: $cp \in \mathcal{C}$. We increment $sup(cp)$ by 1. At the end, we remove from $\mathcal{C}$ those cadence-pairs whose supports are below the parameter τ_{cp} and return the resulting $\mathcal{C}$ as the set of mined frequent cadence-pairs.

Frequent Trend-Pairs Mining. It is to mine the set of all frequent trend-pairs, denoted by $\mathcal{TP}$. This task can be done by a frequent subsequence/substring mining algorithm [8]. Due to page limit, we omit the details here.

2) Phase II: Melody Composition: In this phase, we compose a melody based on the inputted lyric by using the mined frequent patterns in Phase I.

Let M be the sequence of pitches to be composed. We have four steps which we introduce next.

Step 1: Tone Trend Extraction. This step is to extract the tone trend of the inputted lyric, which could be done similarly as we extract the tone trend of the lyric of each song in $\mathcal{D}$ (in Section II-B1). Let T be the resulting tone trend and $n = |T|$ (i.e., there are $n+1$ tones in the lyric and thus there are $n+1$ pitches to be composed in M).

Step 2: Cadence Part Composition. This step is to compose the cadence part of the melody (i.e., $M[n:n+1]$). The main idea is to utilize the correlation information between cadences and the ending tone differences, which is captured by the set of frequent cadence-pairs $\mathcal{C}$. Specifically, we select a cadence from a "cadence pool" $\mathcal{CAD}$ probabilistically to act as the cadence part of the melody. $\mathcal{CAD}$ is a set of cadences in all cadence-pairs cp in $\mathcal{C}$ whose tone differences are the ending tone difference in T (i.e., $T[n]$), i.e.,

$$\mathcal{CAD} = \{cp.cad | cp \in \mathcal{C}, cp.toneDiff = T[n]\}$$

Besides, we associate with each cadence $cad \in \mathcal{CAD}$ a support, denoted by $sup(cad)$, which is set to be $sup(cp)$ where cp is its corresponding cadence-pair in $\mathcal{C}$ (i.e., $cp.cad = cad$ and $cp.toneDiff = T[n]$). Let cad^* be the cadence part of the melody to be composed, i.e., we have $M[n] = cad^*[1]$

and $M[n+1] = cad^*[2]$. We select a cadence from $\mathcal{CAD}$ as cad^* by using the following probability distribution.

$$Pr(cad^* = cad) = \frac{sup(cad)}{\sum_{cad' \in \mathcal{D}} sup(cad')} \quad (1)$$

Step 3: Non-Cadence Pitch Trend Construction. This step is to construct the pitch trend of the non-cadence part of the melody (i.e., $M[1:n-1]$), denoted by $P[1:n-1]$. The main idea is to construct $P[1:n-1]$ (which is a pitch trend) based on $T[1:n-1]$ (which is a tone trend) by utilizing the correlation information between the pitch trends and the tone trends of the songs in $\mathcal{D}$, which is captured by the set of frequent trend-pairs $\mathcal{TP}$.

Specifically, we construct $P[1:n-1]$ sequentially *from the end to the beginning* with an iterative process (the reason for this reverse chronological order is that the cadence which is at the end has been composed already (at Step 2) and starting from the cadence makes the melody smooth). We maintain a variable end which indicates that $P[1:end]$ has not been constructed yet during the process. At the beginning, end is initialized to be $n-1$. Then, we proceed with iterations. At each iteration, we have four steps.

First, we construct a "pitch trend pool" $\mathcal{P}_{pool}$ which corresponds to a set of pitch trends that could be used to construct a suffix of $P[1:end]$ (i.e., $P[i:end]$ where $1 \leq i \leq end$). $\mathcal{P}_{pool}$ contains the pitch trends of those frequent trend-pairs in $\mathcal{TP}$ each of which has its tone trend *matching* a suffix of $T[1:end]$ (i.e., $T[i:end]$ where $1 \leq i \leq end$). Formally,

$$\mathcal{P}_{pool} = \{tp.P | tp \in \mathcal{TP} \text{ and } \exists i \in [1, end] \text{ such that } tp.T = T[i:end]\} \quad (2)$$

Besides, we associate with each P' in $\mathcal{P}_{pool}$ a support, denoted by $sup(P')$, which is set to be the support of its *corresponding* trend-pair in $\mathcal{TP}$, i.e., $sup(P') = sup(tp)$ where $tp \in \mathcal{TP}$ with $tp.P = P'$ and $tp.T = T[i:end]$ for a $i \in [1, end]$.

Second, we pick a pitch trend P^* from $\mathcal{P}_{pool}$ falling the probability distribution as defined in Equation 3.

$$Pr(P^* = P') = \frac{sup(P')}{\sum_{P'' \in \mathcal{P}_{pool}} sup(P'')} \quad (3)$$

Third, we specify the portion of $P[i:end]$ to be P^* where $i = end - |P^*| + 1$.

Fourth, we update end to be $i-1$. We stop the above process if $end = 0$ and continue to the next iteration otherwise.

At the end, $P[1:n-1]$ has been constructed completely.

Step 4: Non-Cadence Part Composition. This step is to compose the non-cadence part of the melody (i.e., $M[1:n-1]$). It is done by using the pitch trend $P[1:n-1]$ constructed at Step 3 and Equation (4) (by definition of a pitch trend).

$$M[i] = M[i+1] - P[i] \ for \ i = n-1, n-2, ..., 1 \quad (4)$$

Finally, the melody (i.e., $M[1:n+1]$) is composed completely.

III. Software

We developed a software with the Trend-MC algorithm as its core, which is called *Trend-MC* as well for convenience.

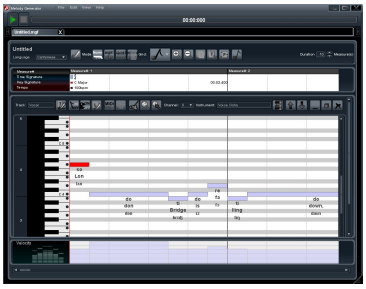

Fig. 2. The main interface of Trend-MC

A. User Interface

The main interface of Trend-MC is shown in Figure 2. Trend-MC supports a rich set of functionalities related to melody composition. Some examples include lyric management (e.g., alignment, input and tone exaction), music parameter specification (e.g., key signature, time signature and instrument), melody management (composition, edit, play and representation) and some other utilities that provide many options of displaying, playing and editing melodies/lyrics. Some of these functionalities would be illustrated in the demonstration part in Section III-B.

B. Demonstration

We shot a video of using Trend-MC to compose a piece of melody based on a prepared lyric, which demonstrates many of the functionalities of Trend-MC. The composed melody sounds nice. This demo could be found at *http://www.cse.ust.hk/~raywong/demo/demo.avi*.

IV. Conclusion

In this paper, we proposed a new idea of capturing the correlation information between melodies and lyrics by trend-based frequent patterns. Based on this idea, we designed a melody composition algorithm called Trend-MC which avoided the shortcomings of previous methods. Besides, we developed a software with the Trend-MC as its core.

Acknowledgement

We are grateful to the anonymous reviewers for their constructive comments on this paper. The research was supported by VPRGO15EG10.

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] C. Long, R. C.-W. Wong, and R. K. W. Sze, "T-music: A melody composer based on frequent pattern mining," in *ICDE*, 2013.

[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. O. Rabin, "Probabilistic automata*," *Information and control*, vol. 6, no. 3, pp. 230–245, 1963.

[8] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," in *KDD*, 2002.