Automatic Synthesis of Notes Based on Carnatic Music Raga Characteristics

Janani Varadharajan, Guruprasad Sridharan, Vignesh Natarajan, Rajeswari Sridhar,
Department of Computer Science and Engineering
College of Engineering, Guindy

Abstract—In this paper, we propose two methods to automatically generate notes (Swaras) conforming to the rules of a given Raga in Carnatic music. The proposed methods require training examples for learning the probability model of the chosen Raga and no hand-coded rules are required. Since the methods are purely statistical in nature, it is easy to extend them to work with a large number of Ragas. Each proposed method involves a Learning Phase and Synthesis Phase. In the Learning Phase, an already existing composition of the chosen Raga is used to learn the transition probabilities between Swara sequences based on the Raga lakshana characteristics. In the Synthesis Phase, using the previously constructed transition matrix, Swaras are generated for the desired Raga. We describe two methods - one using First Order Markov Chains and the other based on Hidden Markov Models. Both the methods are validated based on feedback from experts trained in Carnatic music.

Keywords—Carnatic Music, Markov Chain, HMM, Sequence Models.

I. Introduction

A. Terminology

Swara - A Swara is the fundamental unit of Classical Music [1]. A Swara can be analogously understood as a musical note. The basic Swaras in Carnatic music are seven in number and they are as follows: Sa, Ri, Ga, Ma, Pa, Da, Ni.

Raga - A Raga is an arrangement of notes in a predefined manner (musical scale) where the position and neighbourhood of each Swara(note) depends on the Raga lakshana of a Raga. The Raga lakshana includes Aarohana (ascending arrangement of notes), Avarohana (descending sequence of notes), the Amsa (characteristic phrase), the frequently used starting note, most frequently used note, ending note, note not used, etc.

Aarohana - Aarohana, Arohana, Arohanam or Aroha, in the context of Indian classical music, is the ascending scale of notes in a Raga. The notes ascend in pitch from the lower tonic towards the upper tonic.

Avarohana - Avarohana, Avarohanam or Avaroha, in the context of Indian classical music, is the descending scale of any Raga. The notes descend in pitch from the upper tonic down to the lower tonic, possibly in a crooked manner. In our work, we use the Raga lakshana like Aarohana, Avarohana and Amsa (Characteristic phrase) of each Raga as features

to construct a stochastic transition model which is used for synthesis.

Melakartha - A collection of fundamental ragas (musical scales) in Carnatic music (South Indian classical music). Melakartha ragas are parent ragas from which other ragas may be generated.

B. Fundamental Idea

The Ragas are defined not only by the different Swaras that are used to compose it, but also the order of occurrence of these Swaras and one or more characteristic Swara phrases occurring frequently in the composition, thereby giving the Raga an unique signature or an identity. Thus, it is not possible to simply use permutations and combinations of different Swaras to compose a melody in a particular Raga. For instance, the Ragas **Shankarabharanam** and **Kalyani** are both Melakartha Ragas, meaning they have all seven basic Swaras in their *Aarohana* and *Avarohana*.

Shankarabharanam:
Aarohana : S R_2 G_3 M_1 P D_2 N_3 S
Avarohana : S N_3 D_2 P M_1 G_3 R_2 S

Kalyani:
Aarohana: S R_2 G_3 M_2 P D_2 N_3 S
Avarohana: S N_3 D_2 P M_2 G_3 R_2 S

If mere permutations and combinations of Swaras are used to generate compositions in these Ragas, one would not be able to easily identify which Raga a particular composition follows, if the compositions do not use the Swara M. The characteristic phrases must be accentuated well enough in the composition to enable correct identification of the Raga.

In this paper, we describe an algorithm that learns the characteristics of Ragas and synthesizes sequences conforming to the properties of that particular Raga. The primary motive of this paper is to demonstrate the ability of a probabilistic algorithm to understand the characteristics of a Raga and reproduce sequences of the same Raga that do not delve from the rules of the said Raga.

This paper is organized as follows: Section 2 talks about the related work in this field, Section 3 explains the proposed method, Section 4 discusses the assumptions and constraints

that exist in the method proposed, Section 5 talks about the results and Section 6 concludes the paper with possible future extensions.

II. Related Work

Sahasrabuddhe in [3] talks about generating automatic and computer assisted notes that correspond to a particular Raga by constructing a Finite State automata for the Raga, with each states in the Finite State Automata being the alaps (or) phrases of the Raga. The paper claims that remembering the last two notes can produce reasonable performance in a majority of ragas. The authors have further stated that preserving information about elongation, grace notes, and/or successor selection frequencies often enhances the quality of performance.

Dipanjan Das et al in [4] investigated the generation of Aarohanas and Avarohanas present in each Raga that conform to the Hindustani Classical Music for the purpose of discovering and identifying new sequences. This idea implements a Finite State Machine (FSM) which was proposed by [3] that generates a sequence of Swaras that conforms to the rules of a particular Raga. In [4], the authors designed a probabilistic finite state machine that generates only the Avarohana and Aarohana of the said Raga. The FSM used in the paper can be looked upon as a bigram model of three most frequent notes along with their probabilities that they follow while moving in either direction so as to generate the ascending and descending sequences of the particular Raga. The FSM for each Raga was constructed manually as the idea deals with only three most frequently occurring notes. Once the FSM created the ascending and descending notes, an algorithm is run on it to generate one instance or sample of the said Raga. This algorithm takes the number of notes and the start note of the Raga as input. Output of the entire process would be a sequence conforming to a particular Raga which was generated upon constructing a sample of the Aarohana and Avarohana of the same. The main drawback that we have identified from this paper is that, a sequence generated using a randomized approach without any defined rules to monitor the occurrence and position of the notes will result in a sequence that does not conform to the characteristics of the Raga.

Subramanian in his landmark paper [5], talks about generating computer music from skeletal notations, along with the gamakkas that produce a smooth transition from one Swara to another. The proposed system uses user defined gamakkam definition files for each Raga. Addition of gamakkas depend on a) The Raga b) The context in which the note occurs and c) The duration of the note. Gamakkams can be modeled in different ways. One method would be to analyze a large number of recordings sung by music experts and extract common features for each note of the Raga. This requires a system that can identify note boundaries and transcribe live music into notation format.

III. Method

The method to generate notes for a Raga is given in Figure 1. The following subsections discuss the proposed methods to generate Raga notes.

A. First Order Markov Chain

1) Learning Phase: The idea of using Markov chains was inspired by the work of Simone Hill in [6]. The learning phase involves the computation of the probability with which the Swaras follow one another. Note sequences (transcripts) of existing compositions of the chosen Raga are input to the system in the form of text and the algorithm constructs a 16*16 (seven basic Swaras along with their subtypes) state transition matrix for the Raga based on this sequence. This matrix stores the state transition probability from one Swara to another in a sequence. By this way, frequent bigrams of the chosen Raga are assigned high probabilities.

2) Synthesis Phase: In the synthesis phase, a sequence of Swaras pertaining to the desired Raga is generated based on the probabilities in the transition matrix of the Markov Chain. Samples are drawn using a Pseudo Random Number generator based on the cumulative probability for transitions from each input Swara. The initial Swara is chosen based on a starting probability distribution estimated using multiple compositions of the same Raga. As a result of sampling according to Markovian distribtuion, we get a sequence of Swaras of desired length.

3) Text to Music Phase: In this phase, each of the sixteen Swaras is assigned a frequency relative to a chosen fundamental frequency based on the rules enforced by the Raga (scale). The composition generated in the previous phase is fed into this phase which then maps the Swaras in the composition to its corresponding frequency and subsequent generation of music is done using Matlab with Karplus-Strong string synthesis algorithm.

B. Hidden Markov Model

Hidden Markov Models (HMMs) are used to model finite discrete or continuous observation sequences when the state transitions cannot be observed directly [7]. HMMs have been employed successfully for speech recognition, speech synthesis, machine translation, parts-of-speech tagging, cryptanalysis, gene prediction, etc. HMM is an extension to the finite state Markov model which makes it more applicable to real-world processes, where the sequence of states cannot be observed. HMMs have been used for generation of chord harmonizations in Western Music [6]. The ability of HMMs to capture complex harmonies from the training sequence makes it attractive for generation of melodies. We propose to use HMM for generation of Carnatic music and provide a comparison of its performance against the method previously discussed, in the Results and Analysis section.

1) Algorithm: The algorithm takes as input a Raga chosen by the user and a training sequence. The training sequence consists of Swaras which constitute a musical piece conforming to the chosen Raga. The algorithm learns from the training sequence and outputs a sequence of generated Swaras conforming to the chosen Raga.

Fig. 1: Block Diagram of the notes synthesis system
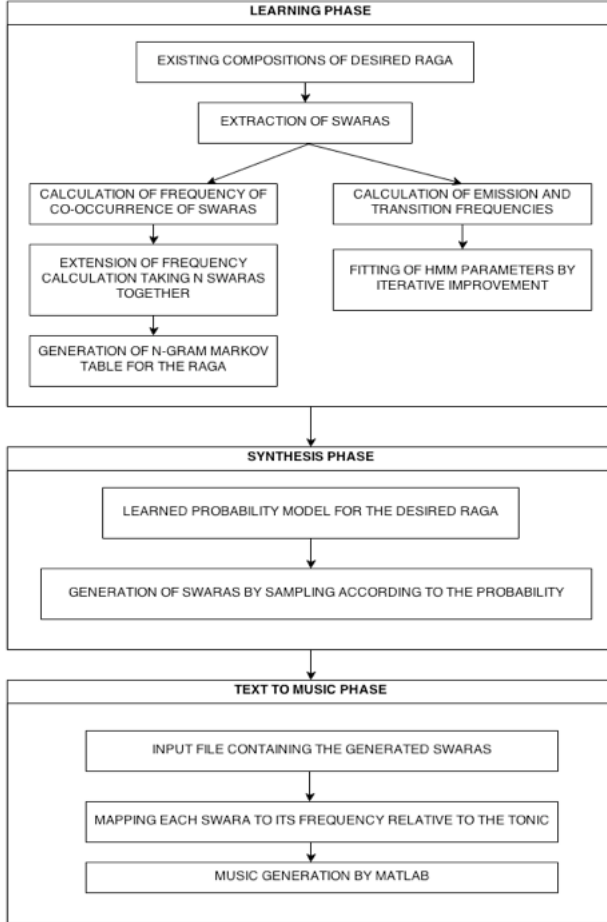
The Hidden Markov Model is composed of three probability distributions namely Transition, Emission and Initial probability distributions. The parameters include number of hidden states and visible states, and the vocabulary used(set of symbols emitted).

We use a finite discrete HMM model for generating the sequence of Swaras. The hidden states correspond to the characteristic phrases (a group of Swaras). The emitted symbols correspond to the Swaras.

The intuition behind using HMMs is that they are better at capturing this phrasal structure of Swaras from the training examples which is necessary for generating complex and more nuanced harmonies.

HMMs are typically described using a three-tuple representation [6] $M = (A, B, \Pi)$, where
A - $N \times N$ matrix which describes transition probability between states.
B - Emission probability distribution ($B = b_1, b_2, \ldots, b_N$).
- Initial probability distribution over states.
N - Number of hidden and visible states.
a_{ij} - Transition probability between i^{th} state and j^{th} state.
b_i - Emission probability for the ith state.
X_t - Random variable which denotes the state of the system at time t.
Y_t - Random variable which denotes the symbol emitted at time t.

The aim of the learning phase is to find a model M in the model space M^* such that

$$M = \underset{M \epsilon M^*}{\mathrm{argmax}} \, P(Y|M)$$

Baum-Welch algorithm, a specialization of *Expectation-Maximization algorithm* is used for learning the model. It learns that model which maximizes the likelihood of observing the sequence training sequence Y.

Number of hidden states were chosen by a cross-validated approach. Scikit-learn package for Python provides an implementation for categorical HMM [9][10] which was used for the implementation and evaluation.

Steps:
1) Fetch the training sequences for the chosen Raga from the database.
2) Start with some random initialization values for the 3-tuple.
3) Apply forward and backward passes of the Baum-Welch algorithm on all the training sequences to update the conditional model variables for achieving Expectation-Maximization and repeat until convergence based on threshold T.
4) Repeat the algorithm multiple times with different initialization values and find the model with the best score.
5) Sample from the trained HMM model to henerate output sequence of desired length with starting symbol (*Swara*) chosen by sampling from initial distribution derived from training sequences

IV. LIMITATIONS

- The proposed system does not handle the gamakkas of the generated composition.
- The system doesn't try to fit tala (time singature) to the generated composition.
- The system proposed is purely statistical in nature so the quality of the generated composition is limited by the richness of corpus data used in the learning phase.

V. RESULTS AND ANALYSIS

The aim of the experiment was to measure the goodness of the notes generated by the generative model and to validate whether the notes correspond to the Raga given as input.

Six people trained in Carnatic music took part in the experiment. A dataset consisting of songs in .wav format was prepared. The participants were asked to rate each musical composition in the dataset on a scale of 1 to 5, based on their perceived similarity to the corresponding Raga to which the composition originally belonged.

TABLE I: Table showing the mean and standard deviation (enclosed by parantheses) of scores on each Raga for both the methods

Raga	Hidden Markov Model	First Order Markov Chain
Thodi	1.75 (0.41)	1.5 (0.45)
Shankarabharanam	2.5 (0.63)	2.5 (0.63)
Mohanam	3 (0.45)	2.75 (0.69)
Malahari	2.25 (0.52)	1.75 (0.69)
Kalyani	2.5 (0.71)	2.5 (0.55)

VI. CONCLUSION

In this paper, we have compared the performance of First Order Markov Model with that of HMM for synthesis of Carnatic Music. The result indicates that even though HMM performs better than Bigram model, the performance is poor on an absolute scale. Future work can focus on the evaluating the use of discriminative models such as Conditional Random Fields for this problem. It is also worthwhile to explore the use of sequence statistics to construct a scoring function for automated scoring of the generated compositions.

ACKNOWLEDGMENT

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Sambamoorthy, Pichu. South Indian Music. Vol. 4. Indian Music Publishing House, 1969.

[3] H. V. Sahasrabuddhe, *Analysis and Synthesis of Hindustani Classical Music*, University of Poona, 1992.

[4] Das, Dipanjan, and Monojit Choudhury. *Finite state models for generation of Hindustani classical music*. Proceedings of International Symposium on Frontiers of Research in Speech and Music, 2005.

[5] M. Subramanian, *Generating Computer Music from Skeletal Notations for Carnatic Music Compositions*, Proceedings of the 2nd CompMusic Workshop, Istanbul, Turkey, July 2012.

[6] Simone Hill, Markov Melody Generator, 2011.

[7] D. Steinsaltz, and D. Wessel. In Progress, *The Markov Melody Engine: Generating Random Melodies With Two-Step Markov Chains*. Technical report, Department of Statistics, University of California at Berkeley.

[8] C. Kohlschein. *An introduction to hidden Markov models*. Probability and randomization in computer science, seminar in winter semester, Aachen University, 2006/2007.

[9] Pedregosa, Fabian, et al. *Scikit-learn: Machine learning in Python*. The Journal of Machine Learning Research 12 (2011): 2825-2830.

[10] Oliphant, Travis E. *Python for scientific computing*. Computing in Science & Engineering 9.3 (2007): 10-20.