Byzantine Music Composition Using Markov Models

Teodor Lucian Anton
University Politehnica of Bucharest
313 Splaiul Independentei,
Bucharest, Romania
teodordanton@gmail.com

Stefan Trausan-Matu
University Politehnica of Bucharest
313 Splaiul Independentei,
Bucharest, Romania
and
Research Institute for Artificial Intelligence
and
Academy of Romanian Scientists
stefan.trausan@cs.pub.ro

ABSTRACT

This paper presents a project that applies Markov Models improved with similarity metrics to produce computer-generated Byzantine music. An implementation over a dataset of Romanian music in eighth mode, stichiraric style is described. The importance of the similarity mechanism is highlighted, in particular in regard to melodic transition tables. Results of the implementation are presented, which show both strengths and limitations of the model.

Author Keywords

Music composition; Markov Models; Byzantine Music; rhythm

ACM Classification Keywords

H.5.1. Multimedia Information Systems: Audio input/output; Applied computing → Sound and music computing; Computing methodologies → Artificial intelligence; Machine learning;

INTRODUCTION

Advanced interfaces should include all communication means that can enhance human-computer interaction. In this sense, music has an important role, being able to provide an adequate context for accompanying various tasks, for immersing the user in a specific atmosphere. Therefore, generating/composing music with the computer in the aim of facilitating immersion should be able to provide various specific melodicities. There already are an important number of computer programs that generate several common music genres: jazz, rock, pop, blues, classical music (for example, in Bach's style), using various techniques, such as: rule-based systems, constraint satisfaction, neural networks, evolutionary programming, Markov Models, etc. [3] These programs may generate music from scratch starting from some rules (written by the programmer or obtained by machine learning) or by "imitation", through supervised machine learning, Markov Models being very well suited for temporal sequences of events, specific also to music. Another way to generate music is the sonification of processes that have musicality [6].

However, at least as far as we know, there are no computer programs that generate Byzantine or Gregorian music, even if there are many contexts in which these kinds of music may accompany human-computer interaction, for example in virtual museums, when reading specific texts, or even as a general background music.

This paper mainly contains the results of the research and implementation done by the first author for his graduation thesis, which expand upon Ebert's ideas of a phenomenological approach to artificial jazz improvisation [2], testing them out on a very different kind of music: the Byzantine chant without ison. Ebert declares his approach 'phenomenological', thus, inspired by the processes of the human mind. He notes that jazz improvisation is based less on well-defined musical laws and more on the players' experience with listening and improvising on a given musical theme. Ebert mentioned that an important component of his algorithm is the addition of similarity metrics, idea implied by the importance of similarity in human creativity [2]. Moreover, a phenomenological philosophy perspective was emphasized as required for human language understanding [7].

Byzantine musical theory, on the other hand, as compared to the liberty of jazz improvisation, is based on a complex set of composition rules which restrain the composer's creativity, without excluding it. There is also a significant difference at a purely musical level, requiring a respective adaptation of the composition algorithm. The groups of notes at the end of each musical breath (known as *cadences*) had to be extracted separately. The similarity metrics (in particular, the rhythmic ones) also had to be changed. Fragment extraction was complicated by the melismata (the notes that are sung for one syllable) that characterise stichiraric chant. The aim of the research was to investigate whether the above-mentioned algorithm is capable of producing music with a Byzantine character, without having explicitly given it any composition rules.

BASIC IDEAS OF THE COMPOSITION ALGORITHM

Generally speaking, the algorithm consists of extracting the most frequent melodic and rhythmic fragments, followed by the calculation of tables containing respectively the probability of one rhythmic fragment following another one ($Pr(r_j|r_i)$), the probability of one melodic fragment following another one ($Pr(m_q|m_p)$), and the probability of a rhythmic fragment and a melodic fragment overlapping ($Pr(r_j,m_q)$), which is computed using the number of occurrences of the extracted fragments in the training set. Then, the probability of a new melodico-rhythmic fragment following any given fragment may be approximated as follows:

$$Pr(r_j,m_q \mid r_i,m_p) \approx Pr(r_j|r_i) \cdot Pr(m_q|m_p) \cdot Pr(r_j,m_q) \quad (1)$$

The similarity metrics are used for extrapolating fore-mentioned probabilities when the fragments in question do not follow each other or overlap at all in the training set. This extrapolation is very important because in practice the transition tables are quite sparse without extrapolation, which narrows the composition possibilities. The extrapolation of unseen probabilities in the rhythmic matrix is done according to the following equation:

$$Pr(r_j|r_i) \approx \kappa_r \cdot Pr(r_j|r_k) \cdot S_{k,i}^{(R)} \quad (2)$$

where k = *argmax(*$S_{k,i}^{(R)}$), κ_r is the rhythmic extrapolation factor that controls the weight of extrapolated transition probabilities relative to directly calculated ones, and $S_{k,i}^{(R)}$ is the similarity between rhythmic fragments r_k and r_i. Extrapolation for melodic transitions and for melodico-rhythmic superposition is done in similar fashion.

The learning process ends with the calculation and extrapolation of the elements of the tables. Afterwards they can be used for generation: a melodico-rhythmic fragment is selected for the beginning, the most likely follow-ups are calculated, one of them is chosen and the steps are repeated, forming a new score. In the case of Byzantine music, a fitting cadence is attached at the end.

DETAILED EXPOSITION

According to the tempo, Byzantine chant is split into four styles, from faster to slower: recitative, heirmologic, stichiraric, and papadic. According to the character of the melody, it is split into eight tones. Every tone has its own set of melodic structures and formulae; in particular, it has a fundamental scale of eight notes, typical cadences and final notes; these are the traits that were looked for in the results of the system. In building the training the used set was only stichiraric, because the recitative and heirmologic styles were found as too simple, and papadic too rare. Of stichiraric chants, only those in eighth tone (whose base scale is in fact C Major) were picked because it is one of the most frequent.

The music used for training was taken from standard Romanian church music books [1,4,5], as it was wished that the set to be as stylistically coherent as possible. The selected chants were transcribed into the score editor *MuseScore*, which was used to convert them to the MusicXML standard annotation language. Most of the secondary elements: legatos, grace notes, lyrics (as a redundant element for checking transcription correctness) were kept, but all dynamics elements were dropped. 17 chants were obtained, for a total of approximately 850 measures.

Every chant is a sequence of *phrases*, and every phrase is a sequence of ‘breaths’. Every breath ends with a cadence - a final formula ending on a very particular note. There are multiple types of cadences, according to the kind of pause that follows them, and every cadence type ends on a certain note of the tone’s base scale; most often, the first or the third or the fifth. For the sake of simplicity, cadences were split into ‘mid-’ - that end a breath inside a phrase, and ‘final’ - that end the final breath of a phrase.

The program was written in Python, which was chosen due to its conciseness and its wealth of useful libraries. It is split into 3 separate phases: data attainment, data processing, and composition.

Data Attainment

Data attainment involves reading the input MusicXML files and generating an internal representation of the data, which is to be stored in serialized form and processed later. Python's *XML ElementTree* API was used to extract the XML tree from the files. The tree was decomposed into individual notes, retaining for each note its duration, pitch and potential legatos with the previous note.

For pitch appreciation the old (and rather complex) Byzantine diastematics were dropped in favour of their Western scale approximation, that is commonly used today in the Romanian Orthodox Church chanting. Having then the Western scale of 12 half-steps, every possible pitch was represented with a number: $G_2\# = -1$, $A_3 = 0$, $A_4 = 12$, etc.

A numeric representation for duration was used as well. Noting that the shortest duration in the dataset is the semiquaver, and that triplets are also present, it was decided upon: ♪ = 6, ♩ = 12, ♩· = 18, etc.

At the end of this phase the data is represented as a list of phrases, with each phrase being a list of breaths, each breath a list of note groups (a.k.a. legatos), each legato a list of notes, each note a pair of step and duration.

Data Processing

Data processing proceeds using the data representation obtained during data attainment to build fragment lists and probability tables. The table set built here is extended compared to that of Egbert [2], since cadences had to be considered as separate fragments. Thus, this phase generates the below data, which are serialized at the end, to be used in the next phase:

1. **R** the list of most frequent rhythmic fragments;
2. **RT** the table of rhythmic transition probabilities;
3. **M** the list of most frequent melodic fragments;
4. **MT** the table of melodic transition probabilities;
5. **AT** the table of melodico-rhythmic alignment probabilities;
6. **TL** the legato translation dictionary;

7. **CR** the list of most frequent cadence rhythms;
8. **CRT** the table of rhythmic cadence transition probabilities;
9. **CM** the list of most frequent cadence melodies;
10. **CMT** the table of melodic cadence transition probabilities;
11. **CAT** the table of melodico-rhythmic cadence alignment probabilities;
12. **B** the list of most likely breath-starting melodico-rhythmic fragments.

In the case of Western musical scores, fragment extraction can naturally be based on the division of the score into measures. Ebert proposes extracting as fragments each measure or each half-measure [2, page 26]. Byzantine music is not divided into measures, and therefore requires a different approach. Fragment extraction was based on rhythm, following the durations of the notes. Multiple fragment lengths were tried, looking for the length that would best align with the durations in the dataset. The conclusion was that the length of one measure (four crotchets, or 48 in the representation of duration of this presented approach) would indeed be the best. A sliding-window procedure was used to extract all available fragments of duration 48, splitting them into rhythmic and melodic components.

During fragment extraction, legatos were considered to be important, since they corresponded to typical Byzantine melismata. For this reason, fragments whose boundaries cut across legatos were not extracted.

At rhythmic fragment selection, it was noted that there were many examples of fragments with the same effective durations, but differently placed legatos, e.g.: [(12) (12 12 12)], [(12 12 12) (12)], [(12 12 12 12)]. They were considered as musically equivalent, but, not wanting to lose essential melismata, the lowest common denominator was computed and **TL,** the legato translation dictionary was created. In the case of the previous example, the respective dictionary entry would be: '[12 12 12 12] : [(12) (12 12) (12)]'. In this way, rhythmic duration was separated from legatos, simplifying the representation and increasing the diversity of the selected rhythmic fragment set, **R**.

A different approach to melodic fragment extraction was chosen: the melodic fragments were classified according to the number of notes. For this reason, the selected melodic fragments set, **M**, is in fact a dictionary with the number of the notes as a key and the list of fragments as the value.

When building the transition tables the exact same approach as Ebert's [2] was used, dividing the number of occurrences of each transition by the total number of transitions.

Rhythmic Similarity

The rhythmic similarity metric was inspired by the Byzantine neumic musical notation, which uses special symbols to halve or double the durations of notes, or to play multiple notes in one duration unit. All fragments were organized into a directed acyclic graph which reflects how splitting durations leads from a fragment to another. The possible splitting operations are:

1. **Binary** Most operations, for example:

   [24] $\rightarrow$ [12 12], [9] $\rightarrow$ [6 3];

2. **Ternary** Only triplets:

   [12] $\rightarrow$ [4 4 4] or [24] $\rightarrow$ [8 8 8].

Then, the distance between two durations would be the minimum number of operations separating them, or the shortest distance between them in the equivalent undirected graph. For example, the distance between [24, 12, 12] and [12, 12, 12, 12] is 1, because only one operation is required (splitting 24 in half); the distance between [6, 18] and [18, 6] is 2, because they are two incompatible splittings of [24].

Using this formalism, it was proven that the shortest distance between two rhythmic fragments can be efficiently calculated using the following algorithm:

1. The two fragments are overlapped and the subfragments that align on duration are identified:

$$\begin{array}{lclll} r_1 = [12, 24, 4, 4, 4] & \multirow{2}{*}{$>$} & [12] & [24] & [4, 4, 4] \\ r_2 = [12, 18, 6, 6, 6] & & [12] & [18, 6] & [6, 6] \end{array}$$

2. For each subfragment in each pair, the *splitting coefficient* κ_{sp} is calculated, that is, the number of splits needed to turn a monolithic duration into that subfragment. For example:

   - For [18 6] κ_{sp} is 1, since only one operation is necessary:

     [24] $\rightarrow$ [18, 6];

   - For [6, 6, 3, 9] κ_{sp} is 3: [24] $\rightarrow$ [12, 12] $\rightarrow$ [6, 6, 12] $\rightarrow$ [6, 6, 3, 9];

   In practice, the splitting coefficient can be calculated using the formula:

   κ_{sp} = *number of durations – number of triplets* – 1.

3. All splitting coefficients calculated earlier are summated; this sum is the shortest distance (*base distance*) between the two fragments. In the case of fragments r_1 and r_2:

   $\Sigma\kappa_{sp} = (0 + 0) + (0 + 1) + (1 + 1) = 3$

An issue with this metric, as with the Hamming distance, is that it cannot detect non-local similarities – that is, patterns that do not align in time. For this reason, the base distance calculation was combined with the *permutation* of the second rhythm. A additional cost for permutation was set to the number of inversions. Then, the *effective distance* $D_{r1,r2}$ is the minimum of distances between all permutations. Similarity is calculated using a function that converts distance into coefficient [2, page 31].

$$S_{r1,r2}=1/\sqrt{1+D_{r1,r2}} \quad (3)$$

Melodic Similarity

The melodic similarity metric is based on the *geometric distance* [2, page 33]. It can be proven that the distance between two fragments, as a function of vertical shifting, has a convex graph, fact which leads to a more efficient search for the smallest distance. The similarity coefficient calculation is also done using formula (3).

These similarity metrics and formula (2) are used to calculate the final form of the transition tables **RT**, **MT**, and **AT**. Similarly, the cadence was calculated in transition tables **CRT**, **CMT**, **CAT**, which give the probability of a normal fragment transitioning into a cadence fragment.

Composition

The score generation phase starts with the deserialisation of the tables obtained at the previous phase, followed by the selection of a breath-starting fragment. The fragments most likely to follow after the current fragment are calculated, and one is picked at random. This is repeated a specified number of times, after which a suitable cadence is selected. Then the generated sequence is converted into MusicXML and stored in a new file.

RESULTS

The cutting out of cadences during cadence extraction showed to lead to a significant decrease of data available for ordinary fragment extraction and transition calculation, due to the shortness of the chant's breaths. This meant it was needed to add extra scores to the training set. Due to the variety of melodic fragments, melodic transition tables proved to be very sparse: out of 7056 elements, only 431 (6.1%) were non-zero. An additional 499 transitions were extrapolated using the similarity metric, more than doubling the number of non-zero elements and therefore greatly expanding melodic evolution possibilities.

It also was noted that some transitions were far more prevalent than others, even when applying thinning to the transition tables [2, page 29]. This made the generated material rather predictable, often leading into a cycle after 4-6 measures. This was improved by introducing a random element into the selection process.

Evaluation

The music was evaluated using principles described in [8]. Considering the three examples in Figure 1: first two exhibit progressions across the subscale C–G and cadences in G and C, in accordance with the rules of the first species of eighth mode chant [8, page 29 & 366]. The third shows melodic evolution on the tetrachord of F–B♭, in accordance with the rules of the second species of eighth mode chant [8, page 29 & 377]. A notable aspect of the third example is the exclusively syllabic character (all crotchets, no legatos), whereas stichiraric style requires more ornamentation.

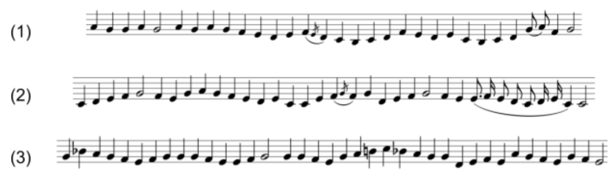

Figure 1: Some melodies created by the system

Overall, the tone of eighth mode stichiraric is described as serene, imposing, and even majestic in mood [Giu81, page 364]; the generated sequences were found to be serene and dignified, which is generally consistent with the description of the music species.

CONCLUSION

The algorithm successfully learns overall Byzantine characteristics and puts them into music. The main issues seem to be the sparseness of the melodic transition tables, the lack of ornamentation, and the overall ‘aimless’ character of the compositions. A possible remedy for the first would be a different kind of musical fragment extraction, combining similar fragments into one, using *binning* for instance [2, page 27], or using *clustering*. The second issue stems from the great prevalence of the syllabic rhythmic measure, [12, 12, 12, 12], which makes up more than a third of the rhythmic dataset; this could be improved by separating this fragment from the others during fragment selection and learning a separate probability of any measure being syllabic or melismatic. The third issue is due to the limitations of an first-order Markov Model. A solution might be to introduce a parallel system to learn composition patterns at a ‘macro’ level and use it to influence fragment selection during the composition phase.

It is expected that this system could be easily adapted to work with Gregorian chant, given its sisterhood to Byzantine chant.

REFERENCES

1. Cântările Sfintei Liturghii, colinde și alte cântări bisericești. Editura Institutului Biblic și de Misiune al Bisericii Ortodoxe Române, București, 1999.
2. Ebert, M. A Phenomenological Approach to Artificial Jazz Improvisation. In: *Publications of the Institute of Cognitive Science*. University of Osnabrück, 2010.
3. Herremans, D., Chuan, C-H., Chew, E. A Functional Taxonomy of Music Generation Systems, *ACM Computing Surveys*, Vol. 50, No. 5, Article 69, 2017.
4. Lungu, N.C., Croitoru, I., and Costea, G. Anastasimatarul uniformizat: II. Utrenierul. Editura Institutului Biblic și de Misiune al Bisericii Ortodoxe Române, București, 2004.

5. Lungu, N.C., Croitoru, I., and Costea, G. Anastasimatarul uniformizat: I. Vecernierul. Editura Institutului Biblic şi de Misiune al Bisericii Ortodoxe Române, Bucureşti, 2002.
6. Trausan-Matu, S., Calinescu, A. Compunerea de muzică prin sonificarea conversaţiilor chat conform modelului polifonic, *Revista Romana de Interactiune Om-Calculator,* 2015, 8(1), 33-44.
7. Winograd T. Thinking machines: can there be? Are we? Report No. STAN-CS-87-1161, Stanford, 1987.
8. Giuleanu V. Melodica Bizantină. Editura Muzicală, Bucureşti, 1981.