Universal Classification Applied to Musical Sequences

Shlomo Dubnov, Gérard Assayag, Ran El-Yaniv

Universal Classification Applied to Musical Sequences

Shlomo Dubnov
Institute of Computer Science,
the Hebrew University of Jerusalem, Israel

Gerard Assayag
Music Representation Team,
IRCAM, Paris 75004, France

Ran El-Yaniv
Department of Computer Science,
Technion - Israel Institute of Technology

Abstract

In this paper we examine the utility of a certain type of learning techniques that are based on information-theoretic methods for music modeling. Using universal compression algorithms we apply the notion of entropy to characterize sequences in terms of their statistical source coding. This approach provides powerful methods for generation and comparison of sequences without any explicit knowledge of their statistical source. The method was applied for automatic extraction and aleatoric generation of melodies with typical motivic/melodic phrases, style mixture and style classification. The classification results show an interesting grouping that separates works of early classical period from works of a later romantic period.

1 Introduction

Modeling of music is a formidable task that requires understanding of musical phenomena from many aspects, such as music schemata, explicit rules of musical knowledge as well as stochastic aspects of the musical structure. Among the various research methods, one can outline two basic approaches: the knowledge engineering approach whereby one attempts to arrive at explicit formulations of musical rules and schemata based on his musical knowledge and expertise, and another, an empirical "learning" approach, where a statistical model is induced automatically through application of learning algorithms to existing compositions. In this paper we examine the utility of a certain type of learning techniques based on "universal"[1] information-theoretic methods for characterization and classification of sequences. Specifically, the notion of entropy was applied to compare sequences in terms of similarity between their statistical sources. This approach provides powerful methods for comparison and generation of musical sequences without explicit knowledge of their statistical model. One must note that our basic modelling assumption is that musical sequences are realizations of high-order Markovian sources. Markov models were suggested to be useful for music analysis and composition[11],[7]. Among the numerous applications of these methods, let us mention the following:

- Stochastic generation of new sequences that have the same phrase structure as the original sequence, which musically means resemblance on the motivic/melodic level.
- Stochastic morphing: generation of new sequences whose statistics is obtained as a mixture of the original sources (producing a "mutual source"). The extent to which the new sequence is close to one of the original sources or to the other is easily controlled by the mixture parameters. Thus, one can obtain a gradual transition between two sequences x and y, which is correct in the statistical sense.
- Hierarchical classification of the data is obtained by repetitively aglomerating closest sequences.
- Dictionary of parsed phrases can be used for indexing of a MIDI sequence. The significant phrases can be selected according to the probability of their appearance.

[1] The "universality" being in terms of not assuming any a-priory knowledge on the source statistics and having its asymptotic performance as good as any Markov or finite state model.

An interesting application yet to be explored is in the domain of style replication. Statisticial description of the melodic phrases[2], could be combined with higher level musical rules in order to obtain a better modeling of both stochastic and the determinisitc - schematic properties of a musical piece or musical style.

2 On Stochastic Modeling, Prediction, Compression and Entropy

The underlying assumption in our work is that a given musical piece is produced by an unknown stochastic source, and that all musical pieces of the same style[3] are generated by the same set of stochastic sources. We also assume that each of the sources is Markovian of some finite unknown order. Clearly, the Markovian and finiteness assumptions do not allow us to capture arbitrarily complex music structures for there may be, for example, dependency between the first and last notes of an arbitrarily long piece. Nevertheless, by allowing for a sufficiently long dependence on the past, our underlying assumptions can model much of melodic structure in a musical piece.

We make use of the well explored relationship between compression and prediction. This connection is a consequence of the *asymptotic equipartition property (AEP)* (which is the information theoretic analog to the law of large numbers in probability theory). The AEP tells us that if $X_1, X_2, ...$ are i.i.d. random variables distributed according to $Q(x)$, then $-\frac{1}{n}\log_2 p(x_1, x_2, ..., x_n) \rightarrow H(Q)$, as n increases, where $H(Q)$ is the Shannon entropy of Q, $H(Q) = -\sum_x Q(x)\log_2 Q(x)$. For stationary ergodic processes (in particular, for finite order Markov processes) the AEP is called the Shannon-McMillan-Breiman theorem (see [5]). Let $x = x_1, x_2, ..., x_n$ be a random realization of a Markovian source Q over some finite alphabet A. Then, on average, the lower limit on x's compressibility is $H(Q)$ bits per symbol (see [5]). Thus, we can estimate the entropy $H(Q)$ and also the log-likelihood $\log p(x)$ by attempting to compress x using a compression algorithm.

There are two main approaches for compression: dictionary-based methods and model-based methods. The prominent Lempel-Ziv (LZ) algorithms (such as LZ77 [16] or LZ78 [17]) are among the best known dictionary based algorithms and due to their speed and good compressibility, variants of these algorithms are among the most popular compression algorithms to date. The class of model-based algorithms is more general and some members of this class so far achieve the best compression results. Further, model-based algorithms can emulate dictionary-based algorithms while the converse is not true (see [12]). Among the best performing algorithms to date are the model-based compression algorithm such as the *Prediction by Partial Match (PPM)* algorithms [2] and the *Context Tree Weighting* method [15]. In contrast to the (LZ) dictionary-based algorithms these algorithms adaptively build a probabilistic method of the underlying unknown source given the prefix of the sequence to be compressed. (The dictionary-based methods do not directly count events and estimate probabilities.)

A *universal* compression algorithm is an algorithm that asymptotically achieves the best possible compression rate (i.e. the entropy of the source) for any stochastic source. Both the dictionary-based compression algorithms and the model-based algorithms have members which are universal. Both classes of compression algorithms can be used to estimate a stochastic source and therefore, can be used to synthesize random (music) sequences. Our preliminary attempts to generate music using the PPM algorithm where not successful and we achieved better results with the dictionary-based LZ78 algorithm.

3 Generation of melodic sequences using Lempel-Ziv algorithm.

The Lempel-Ziv data compression algorithm [17] uses an efficient one-pass pattern detection mechanism in order to build its dictionary of substrings. For the purpose of sequence generation, we ignored the encoding part of the algorithm, but utilized its pattern detection and representation scheme. We found this parsing and representation method extremely fruitful for the purpose of statistical generalization/prediction, i.e. deriving new musical sequences from existing phrases on statistical basis.

In our experiments we used either monodic midi streams, or extracted significant voices from polyphonic midi recordings. In certain cases, where the harmony was distributed sequentially and not simultaneously between voices (e.g. Sixth Invention by J.S. Bach), we used the whole polyphony. The time dimension is neutralized. Our input (and output) data is then just a stream of pitches encoded into Midi eight-bits numbers.

[2] Such phrases are sometimes termed melodic "gestures"

[3] Style could be related to a specific composer, genre of the piece or style of an era

The LZ parsing algorithm parses a sequence sequentially into distinct phrases, such that each phrase is the shortest string which is not a previously parsed phrase. From the Lempel-Ziv dictionary, we derive another representation, that we call Lempel-tree. It is convenient to consider this procedure as a counting process [9], where the outcomes are sorted into bins. Each outcome $x(n)$ is classified into a bin determined be a string that starts at the beginning of the current phrase and ends at $x(n-1)$. The first bin, labeled by the empty string, contains all the symbols that appear in the beginning of a phrase. With each new phrase, a new bin labeled by that phrase is added, so that for a sequence $x(1), x(2), .., x(n)$ that was parsed into c strings, we obtain $c+1$ bins. Each bin contains now all possible next outcomes for a string whose immediate past corresponds to the bin label.

We have implemented our experiment in OpenMusic, a visual programming language for music composition and analysis developed at Ircam [1]. OpenMusic has proved to be a very practical environment for specifying and experimenting with this kind of algorithms. As can be seen in the patch window in Figure 1 the process is divided into several steps :

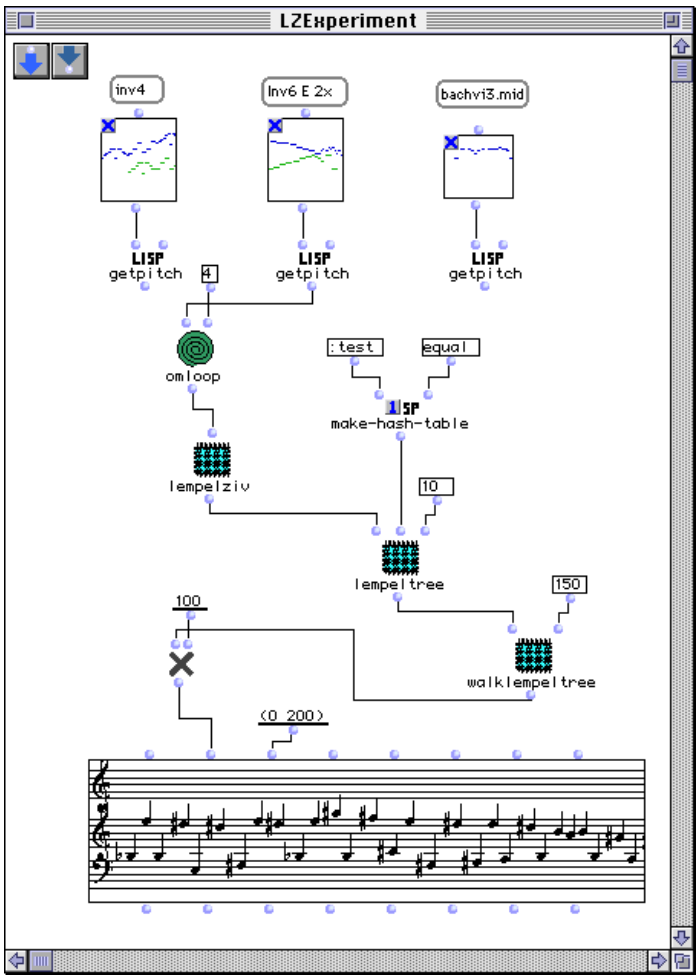

Figure 1: OpenMusic patch window containing the different subpatches that implement the Lempel-Ziv sequences generating algorithm.

1. Extract the pitches from the Midi file.
2. Build the Lempel-Ziv table (subpatch lempelziv)
3. Build the Lempel-tree (subpatch lempeltree)
4. Traverse the Lempel-tree in a random way (subpatch walklempeltree)

5. Send the resulting pitch list into a note-sequence OpenMusic object.

The control parameters for this experiment are the number of time the input sequence is repeated, the maximum length Lp of the prefixes in the Lempel-tree, and the maximum length Ls of the sequence generated. On materials like the Bach Inventions, with the restrictions given above, we found that $Lp = 5$ gave interesting variations but with chaotic alternances of harmonic states ; $Lp = 8$ to 10 gave more stable variations that would capture a significant part of the style ; $Lp > 10$ would tend to reproduce integrally long motives from the input. Figure 2 shows an OpenMusic notation editor with the result of a simulation with Lp = 10, based on Bach invention number 6 in E. (note : the notation display in OpenMusic is based on CMN [13]).

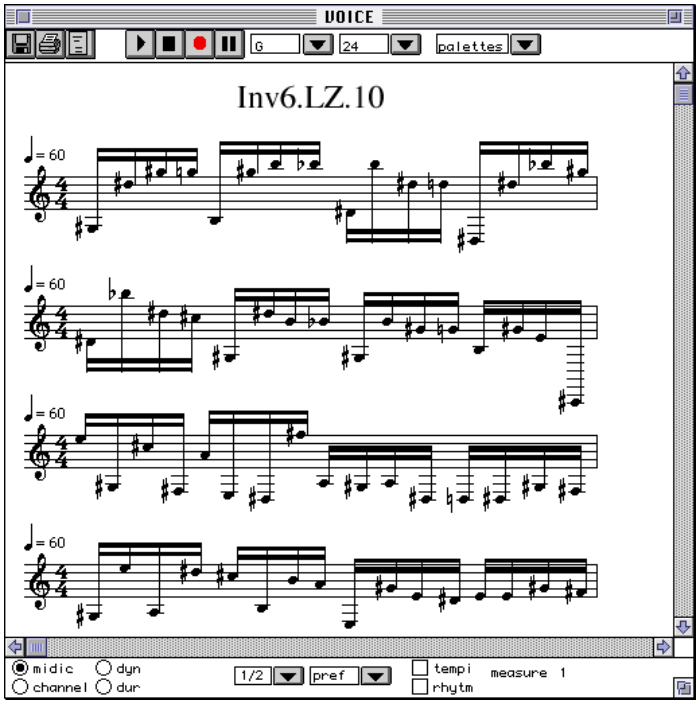

Figure 2: OpenMusic notation editor with the result of a simulation with Lp = 10, based on Bach invention number 6 in E.

4 The sequence mixing algorithm

The above LZ based generation method is effective for generation of a new sequence from a single source. It is not clear though, how to generalize the method to two sources or more (e.g. how to perform a correct mixing or transition (morphing) among stochastic sources). The algorithm described in Figure 3, from [8], generates a new random sequence, given any sample sequences x and y. This new sequence is statistically the *closest* sequence to both x and y in the sense that if S_x and S_y are the sources of x and y, respectively, then the new random sequence is a random realization of the mutual source of S_x and S_y (see Section 5).Notice that this algorithm is completely unparameterized. Also, note that the algorithm is naturally generalizable to any number of sequences.

4.1 Composing Style-Mixture Pieces

The sequence mixing method was used to generate a series of musical works that are "stylistic mixtures" among different composers. The mixing method was applied to pitches and durations separately. To deal with durations, time difference representation was used with note appearance times (note-ons) quantized to 7 duration values. The resulting mixed pitch and duration sequences were combined to produce a new music work.

Initialization:

- $z[0]$ = choose a symbol from x with probability $1/2$ or y with probability $1/2$
- $i = 0$

Loop:

Repeat

- $s_x :=$ max length suffix of z appearing somewhere in x
- $s_y :=$ max length suffix of z appearing somewhere in y
- $\Lambda(s_x, s_x) = \frac{Pr_x(s_x)}{Pr_x(s_x)+Pr_y(s_y)}$
- r = choose x with probability $\Lambda(s_x, s_x)$ or y with probability $1 - \Lambda(s_x, s_x)$
- $r(s_r)$ = randomly choose one of the occurrences of s_r in r
- $z[i+1]$ = the symbol appearing immediately after $r(s_r)$ at r
- $i = i + 1$

End Repeat

Figure 3: The sequence mixing algorithm

CDROM version of the paper contains a recording of three short pieces from "NTrope Suite",[4] an algorithmic composition written using the mixture method. Pairs of works by composers of different styles were used, in the following order: i). Purcel - Grieg, ii). Satie - Hendel, iii). Mozart - Skriabin.

In writing this composition, the computer output was transcribed for a solo recorder, work done in close collaboration with the prominent Early Music musician and a recorders player Drora Bruck[3].

The musical result has successfully demonstrated the capability of the mixing procedure to produce continuous tonal melodies that alternate between different styles. The form of the pieces lacks of course any long term structure, being a result of an aleatoric process.

5 Sequence Matching: Similarity and Entropy

As discussed in the previous section (universal) compression algorithms can be used to generate random music sequences. In addition, they can be used for classification. The universality property is essential since it implies that the algorithm makes minimal assumptions on the underlying unknown source. The general idea is the following. To estimate the (dis)similarity between two sequences x and y one can estimate the *cross entropy* between their underlying sources S_x and S_y. The cross-entropy between two distributions $P(z)$ and $Q(z)$ (also known as the Kullback-Leibler Divergence) is defined to be $D(P||Q) = \sum_z P(z) \log \frac{P(Z)}{Q(z)}$. Although this measure is non-symmetric it is considered to be the standard information-theoretic discrimination measure for distributions. One of the reasons for this is that $D(P||Q)$ measures the inefficiency (in bits) when one attempts to compress a message and assumes that the source distribution is Q while the true source is P. Besides it non-symmetricity, the cross-entropy has an additional, more serious drawback, namely, it is unbounded. Therefore, it is extremely sensitive and unreliable to low probability events under Q.

A different, perhaps more suitable approach is to use the following symmetric and bounded version of the cross-entropy known as the Jensen-Shannon measure [10]. Define the mutual source of P and Q to be $M = \arg\min_{M'} \frac{1}{2}(D(P||M')+D(Q||M'))$, and define the Jensen-Shannon dissimilarity between P and Q, to be $D(P,Q) = \frac{1}{2}(D(P||M)+D(Q||M))$. It is not hard to see that for distributions P and Q this measure uniquely exists (that is, the mutual source is uniquely determined to be $\frac{1}{2}(P+Q)$), and that it is bounded in $[0,1]$. As shown in [8] this measure is related to the following *two-sample problem:* given two i.i.d. samples $x = x_1, x_2, ..., x_{n_x}$ and $y = y_1, y_2, ..., y_{n_y}$, what is the probability that x and y emerged from the same statistical source? As it turns out, this probability is proportional to $2^{-D(P_x,P_y)}$, where P_x and P_y are the empirical probability distributions of the samples x and y. The paper [8] also identifies the mutual source of arbitrary Markovian sources and gives an algorithm to estimate the

[4] This work was presented at the biennale of contemporary music at the Tel-Aviv museum, March 1998.

Jensen-Shannon dissimilarity in this case. This algorithm (described in Section 4) is thus an excellent candidate for computing (dis)similarity between music sequences and we chose to use it here.

6 Composers Data Set and Music Representation

The data submitted to analysis were midi files containing music from the 17th to 20th century [14]. Nineteen (19) composers were selected, ranging from Renaissance and Baroque period to Late Romantic and Impressionist styles. The works were chosen according to the simple criteria of having three works of each composer, picking the shortest works among those appearing in the database. For purpose of classification, the following pre-processing was applied to the data:

1. Polyphonic structure was collapsed into a linear representation, i.e. the polyphonic pitches were transcribed into one line, ordered according to times of appearance[5]. This step was necessary since we had no a-priori knowledge about the voice structure of the piece.

2. To account for pitch range (register) variation, we transformed the music data into a pitch class representation, i.e. modulus 12 of the pitches was used.

3. To deal with differences in tonality, we used the difference representation, i.e. intervals information derived from the pitch class.

4. The time information was not used for classification purposes (accept from the ordering aspect in transcribing polyphonic notes, as described above).

6.1 Classification Results

The distance matrix was computed according to symmetric dissimilarity measure for sequences of [8] (see Section 5). In order to analyze the structure of this data, one typically should apply a data exploratory tool, such as pairwise clustering algorithm. We applied a recent algorithm from [6], that is based on the same Jensen-Shannon dissimilarity measure. The algorithm is iteratively applied to the distance matrix until convergence, separating the matrix into pairs of clusters. The clustering was then applied again to sub-clusters, thus giving a hierarchical pairwise separation. Figure 4 shows the first pairwise clustering result applied to the complete dataset. Black entries in the matrix correspond to distance zero, i.e. for coordinate (i, j) a black entry means a link (similarity clustering) between pieces number i and j. between the pieces The axes (i.e. numbering of the pieces) are roughly placed according to chronological ordering, i.e. approximately corresponding to different stylistic periods. A list containing full names of the music pieces appears in Appendix. One can note a separation into two main groups: the first cluster contains mostly works of an early and classical periods, while the second cluster contains later romantic and impressionistic works.

Figure 5 shows a secondary sub-clustering of the first cluster[6] (i.e. results of applying the clustering procedure to the works in the first cluster). The first sub-cluster contains mostly works of the renaissance and early baroque, while the second sub-cluster contains mostly works of later baroque and classic periods. Four works from the late 19 century appear in the second sub-cluster as well.

6.2 Discussion of the results

Before proceeding to discuss the clustering results, let us state again the objectives of the current study: it was our goal to see how much of the stylistic information can be captured by considering the information in midi streams, without resorting to any musical knowledge. Naturally, one must expect our methods to capture just the most basic music properties related to melodic/harmonic structure of the pieces. Close look at the music works suggest that the main factor that distinguishes between the works is the melodic/contrapunctal character of the early works versus homophonic structure of the later styles.[7]

[5] In case of simultaneous notes (such as chords or simultaneously appearing voices), the pitches were ordered sequentially as well.

[6] The numbers inside the matrix indicate the original piece numbers, i.e. the numbers that appear in the appendix.

[7] It is important to note that comparative clustering according to zero order statistics (i.e. memoryless histograms) did not yield any useful results.

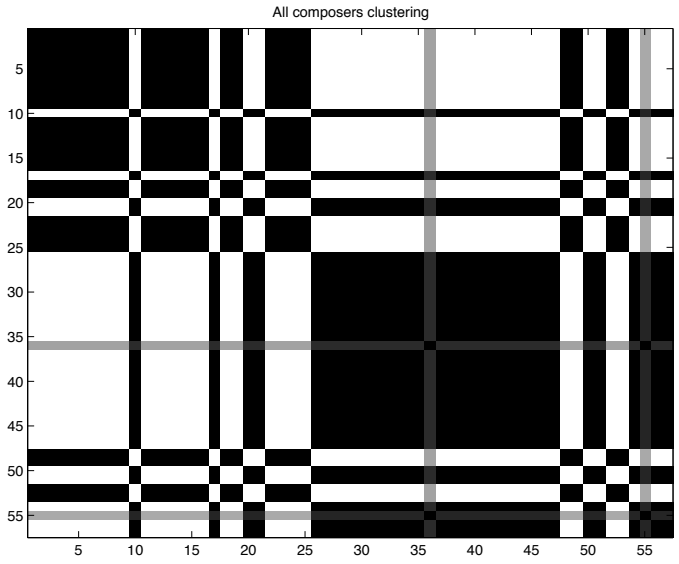

Figure 4: Clustering results of 19 composers, with 3 works by each composer. One can note that the main separation of the two clusters occurs around examples 25-26, which corresponds to works by Beethoven. For discussion of the apparent miss-classifications (8 works) see body of the text.

For instance, the phrases[8] in semitones as parsed by LZ algorithm in Bach's concerto are: 0 9 2 -2 7 5 ; 0 4 0 0 5 9; 0 3 -2 2 0 -3 and etc. It is interesting to notice the "winding" contour of Bach sequences, with step motion balancing jumps, in the opposite direction. In Mozart's Fantasy one finds chromatic scales like : 0 1 2 3 4 5 6 or diatonic scale 0 -1 -3 -5 -6 -8, and other melodic patterns. In contrast to this, Liszt's Liebstrum segments are based on triad structures, such as arpeggio 0 8 3 0 0 8 or sequence of triads 0 7 4 -2 1 7; and etc. Thus, apparently due to "flattening" of the polyphonic structure in the pre-processing stage, the works in the later classic and romantic periods contain mostly triadic sequences. This in contrast to typical sequences of early music pieces that are more of a melodic/motivic character. Considering this interpretation, it is legitimate to consider pieces number 10 or 17 as belonging to the cluster of late music styles, since these examples, although being chronologically and stylistically of early period, have a homophonic structure. Regarding other miss-classified works, there are several pieces that are evidently "mistakes", e.g. works 48, 49 and 52, 53. A careful analysis of the initial distance matrix shows that these examples are extremely noisy. This noise can be related mainly to problems of music representation, which significantly distorts the music data at the pre-processing step, and partly due to noise in the entropy estimates. Since our clustering method is highly non-linear, samples that are initially placed far from either cluster, are bound to be misclassified. Eliminating up to 30% of the noisy data improved drastically the misclassification and removed these examples.

7 Conclusion: Stochastic Models in Music Research

The use of stochastic processes in composition with computers is abundant. Following the pioneering works of Hiller, Xenakis, Koening, Ames, Lorain and others [11], stochastic methods were introduced into modern composition techniques. Although still partly experimental in their nature, these methods allowed composers to consider stochastic phenomena in musical manner, adding a whole new set of acoustic phenomena to the compositional palette. These materials, mostly being "textural" in their nature, are usually based on short time statistics and do not attempt to imitate any melodic or textural materials of existing music. On the other hand, style understanding and style "imitation" usually relies upon knowledge based approaches, i.e. building of complex models that describe the rules of a specific composer, style or an era [4]. Attempts to analyze and statistically generate stylistic materials seem yet to give unsatisfactory results due to the complexity/size and lack of flexibility in the use of statistical

[8] Since the phrases were derived from interval data, they are presented here starting from 0.

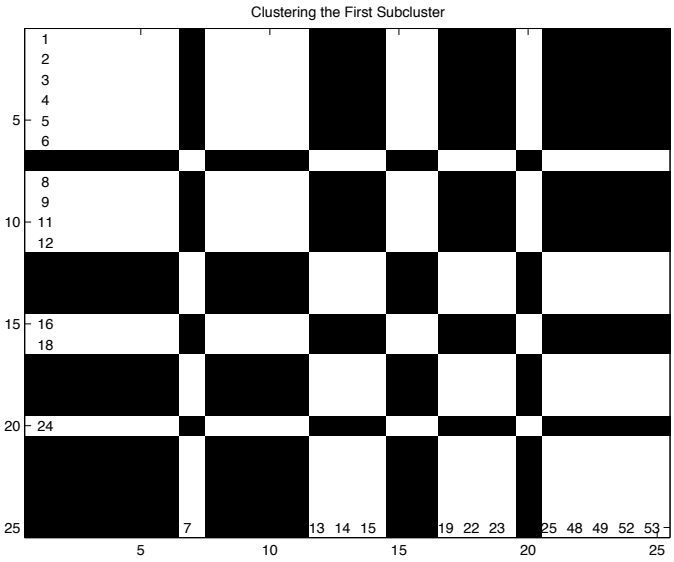

Figure 5: Clustering results of the first sub-cluster. The first sub-cluster contains mostly works of the renaissance and early baroque, while the second sub-cluster contains mostly works of later baroque and classic periods. Four works from the late 19 century appear in the second sub-cluster as well. The numbers inside the matrix indicate the original piece numbers.

models [7].

In the current paper we presented new statistical analyses and re-generation methods that are based upon modern non-parametric techniques of string compression and comparison. These methods are capable of capturing long melodic structures, are easy to implement and give promising results for composition and style classification. Looking at a musical data as a string of pitch or pitch and time-delay events is of course a very poor representation of the actual musical information. On the other hand, limiting ourselves to this type of data, we deliberately eliminated the higher level structure, while still capturing some typical note by note structures. A more careful represention of melodic relations within one voice, and harmonic relations among voices will be considered in future models.

References

[1] Gerard Assayag, Carlos Agon, Joshua Fineberg, Peter Hanappe, An Object Oriented Visual Environment For Musical Composition, Proceedings of the ICMC 97, Thessaloniki, 1997.

[2] T.C.Bell, Cleary, J.P. and Witten, I.H., *Text Compression*, Prentice Hall, NJ., 1990.

[3] Drora's Music Chamber, http://www.actcom.co.il/~dro/

[4] D.Cope, "Computers and Musical Style", A-R Editions, 1991.

[5] T.M. Cover and J.A. Thomas, "Elements of Information Theory", Wiley, 1991.

[6] S. Dubnov, R.El-Yaniv, "New Algorithm for Pairwise Clustering", in preparation, 1998

[7] C.Dodge, T.A.Jerse, "Computer Music", Schrimer Books, 1985.

[8] R.El-Yaniv, S.Fine, and N.Tishby, "Agnostic Classification of Markovian Sequences", To appear in NIPS 1997.

[9] M.Feder, N.Merhav and M.Gutman, "Universal Prediction of Individual Sequences", IEEE Trans. Information Theory 38, 1992.

[10] L.Jianhua, "Divergence measures based on the Shannon entropy", IEEE Trans. Information Theory, 37(1), 145-151, 1991.

[11] K. Jones, "Compositional Applications of Stochastic Processes", Computer Music Journal, Vol. 5, No 2, Summer 1981.

[12] G.G.Langdon , "On parsing versus mixed-order model structures for data compression", IBM Research Report RJ-4163.

[13] B. Schottstaedt, CMN http://ccrma-www.stanford.edu/CCRMA/Software/cmn/cmn.html

[14] R. Schwob, The Classical Midi Archive by Pierre http://www.prs.net/midi.html

[15] F.M.J. Willems, Yu.M. Shtarkov, and Tj.J. Tjalkens, "The Context-Tree Weighting Method: Basic Properties," IEEE Trans. Information Theory, 41(3), 653–664, 1995.

[16] J.Ziv and A.Lempel, "A Universal algorithm for sequential data compression", IEEE Trans. Information Theory 23(3), 337–343, 1977.

[17] J.Ziv and A.Lempel, "Compression of individual sequences via variable rate coding", IEEE Trans. Information Theory, 24(5), 530–536, 1978.

Appendix: Musical Styles Data Set

The following table contains the names of the music works that were considered for the classification task. Due to space limits, we do not specify here the full names of the pieces but rather the shortcuts that correspond to the midi file names, as they appear in [14].

Eyck	Bach	Beethoven	Liszt	Debussy
courant = 1	cantata = 13	baga1 = 25	canzone = 37	clardlun = 49
daphne = 2	chorale = 14	baga2 = 26	liebstrm = 38	deb_ara1 = 50
denavond = 3	concerto = 15	baga3 = 27	valseoub = 39	debptng = 51
Dowland	**Handel**	**Brahms**	**Tchaikovsy**	**Ravel**
fantasia = 4	gfhair = 16	brahm117 = 28	nut0over = 40	ravelpav = 52
lacrimae = 5	gfhgav = 17	brahm118 = 29	nut1mrch = 41	ravelson = 53
queen_el = 6	gfhouv = 18	brahmiz3 = 30	swanlake = 42	ravgleg = 54
Purcel	**Haydn**	**Grieg**	**Musorgsky**	**Satie**
pur_borr = 7	gdance1 = 19	gr_ariet = 31	pm1gnome = 43	gnoss = 55
purmarch = 8	gdance5 = 20	gr_nocte = 32	pmcastle = 44	gymno = 56
purmfaw = 9	gdance6 = 21	gr_waltz = 33	prmchick = 45	gymnop02 = 57
Couprin	**Mozart**	**Chopin**	**Scriabin**	
barimyst = 10	fnt_dmin = 22	chpn_p7 = 34	scriab01 = 46	
coumoiss = 11	minuet6d = 23	chpre1 = 35	scriab06 = 47	
cpf$_b$ird = 12	mozcap = 24	opus28n4 = 36	scriab14 = 48	