A Kind of Bio-inspired Learning of mUsic stylE

Roberto De Prisco, Delfina Malandrino, Gianluca Zaccagnino, Rocco Zaccagnino(✉), and Rosalba Zizza

Dipartimento di Informatica, Università di Salerno, 84084 Fisciano, SA, Italy
zaccagnino@dia.unisa.it
http://music.dia.unisa.it

Abstract. In the field of Computer Music, computational intelligence approaches are very relevant for music information retrieval applications. A challenging task in this area is the automatic recognition of musical styles. The style of a music performer is the result of the combination of several factors such as experience, personality, preferences, especially in music genres where the improvisation plays an important role.

In this paper we propose a new approach for both recognition and automatic composition of music of a specific performer's style. Such a system exploits: *(1)* a one-class machine learning classifier to learn a specific music performer's style, *(2)* a music splicing system to compose melodic lines in the learned style, and *(3)* a LSTM network to predict patterns coherent with the learned style and used to guide the splicing system during the composition.

To assess the effectiveness of our system we performed several tests using transcriptions of solos of popular Jazz musicians. Specifically, with regard to the recognition process, tests were performed to analyze the capability of the system to recognize a style. Also, we show that performances of our classifier are comparable to that of traditional two-class SVM, and that it is able to achieve an accuracy of 97%. With regard to the composition process, tests were performed to verify whether the produced melodies were able to catch the most significant music aspects of the learned style.

1 Introduction

Music is one of the art forms that in many aspects has benefited of the use of computers: sound synthesis and design, digital signal processing, automatic composition, music information retrieval and so on. Computer Music is an emerging research area for the application of computational intelligence techniques, such as machine learning, pattern recognition, bio-inspired algorithms and so on. In this work we face interesting challenges in two different areas: the *music information retrieval* (MIR) and the *algorithmic music composition*. MIR is the interdisciplinary science of retrieving information from music, involving activities such as content-based organization, indexing, and exploration of digital music data sets, while algorithmic music composition addresses the problem of composing music by means of a computer program with no (or minimal) human intervention.

© Springer International Publishing AG 2017
J. Correia et al. (Eds.): EvoMUSART 2017, LNCS 10198, pp. 97–113, 2017.
DOI: 10.1007/978-3-319-55750-2_7

One of the problems to solve in MIR is how to model of *musical styles*. The style of a music performer is the result of his experience and mainly represents his personality or personal preferences. This is even more evident in musical genres where improvisation plays an important role. In this field computer systems could be trained to recognize the main stylistic features of performers. Many works explore the capabilities of machine learning methods to recognize musical style [1–6]. An application of such systems can be their use in cooperation with automatic composition algorithms to guide this process according to a given stylistic profile. In [7] several techniques to define algorithms for music composition have been discussed. In this work we investigate the use of a bio-inspired approach, specifically *music splicing systems* [8].

The main contributions of our paper can be summarized as follows.

1. To propose a machine learning approach for the *recognition* of a music performer's style. We describe a one-class support vector machine classifier to learn musical styles, trained on the significant features extracted from transcribed solos. We show that performances are comparable to traditional two-class SVM, and that the classifier achieves an average accuracy of 97%.
2. To propose a bio-inspired approach for automatic *composition* in a specific music style. We describe a music splicing composer of melodies in the style learned by the classifier. To build the composer we define a Long Short-Term Memory (LSTM) network to predict patterns in the learned style.
3. To assess our approach, we performed several tests by using transcriptions of solos from popular Jazz musicians. We analyzed the capability of our system in recognizing a style and we verified the stylistic coherence of the composed melodies. Moreover, we showed that the use of a LSTM network to predict pattern could involve better results respect to the traditional approaches.

Organization of the paper. In Sect. 2 we describe some relevant related works. In Sect. 3 we provide basic notions about music improvisation and the needed background to understand the paper. In Sects. 4, 5 and 6 we provide details about the machine learning recognizer, the LSTM predictor and the music splicing composer, respectively. Finally, in Sects. 7 and 8 we describe the experiment analysis and we conclude with final remarks and future directions.

2 Related Work

Several works focus on the musical styles recognition problem. Most of these explore the capabilities of machine learning methods. Pampalk et al. [1] use self-organizing maps (SOMs) to cluster music digital libraries according to sound features of musical themes. Whitman et al. [2] present a system based on neural networks and support vector machines able to classify an audio fragment into a given list of sources or artists. In [3], a neural system to recognize music types from sound inputs is described. An emergent approach to genre classification is used in [4], where a classification emerges from the data without any a priori given set of styles. The idea is to use co-occurrence techniques to extract musical similarity between titles or artists. Pitch histograms regarding the tonal pitch are used in [6] to describe blues fragments of Charlie Parker.

We remark that most of these works only take into account the problem of recognizing the styles in terms of music genre, without considering the style of a performer. Furthermore, they do not face the problem of the composition of music in a specific style. Several automatic composers, based on different approaches, have been already proposed in previous works: rules and expert-systems [9–11], systems based on a combination of formal grammars, analysis and pattern matching techniques [12], neural networks [13]. Several automatic composers are based on meta-heuristics, specifically on genetic algorithms [14–20]. Other works have been proposed for different musical genres or problems: thematic bridging [15], Jazz solos [16], harmonize chords progressions [17], monophonic Jazz composition given a chord progression [18], *figured* bass problem [19], and finally, *unfigured* bass problem [20]. In [8,21] the authors define a bio-inspired approach for automatic composition, called music splicing systems. Starting from an initial set of precomposed music the system generates a language in which each word is the representation of music in the style of the chosen composer.

3 Background

We assume that the reader is familiar with the basics of music, machine learning and bio-inspired approaches. However, in this section we provide the basic notations used throughout the paper.

3.1 Improvisation in Music

Music improvisation refers to the ability of playing music extemporaneously, without planning or preparation, by inventing variations on a melody or creating new melodies. In this work we will use the term *solo* to indicate the transcription of an improvised melody. There are many musical genres in which the improvisation assumes a fundamental role during a performance. Usually the improviser's choices are conditioned by own personal experience or preferences, and also by the specific period in which music is collocated. It is interesting to notice that music performers of the same period used similar music choices during their improvisations. Furthermore, during the improvisation, musicians often borrow and customize existing musical pieces from previous performances (also for other musician's performance). Despite this, in each music performer it is possible to found music features that characterize his specific style, and an expert ear is able to perceive such aspects. Thus, given a corpus of solos, we can extract such significant features and using them to recognize his style. We remark that in our approach we are interested in modeling the style of a musician by looking at his music performances and not at the music genre or the music period to which the performer belongs to.

One of the key for extracting the significant features is to study the role that each music note assumes in the specific context (chord) in which it is played. Formally, let Ch be a music chord, $S(Ch)$ be a scale (sequence of notes) chosen by the musician and n be a note played on Ch. In this work we assume that each scale is composed of 7 notes. Then it is possible to associate a degree (position) to n respect to $S(Ch)$. We indicate with $Degree(n, S(Ch))$ such a position. The degrees are indicates with roman numerals, and obviously there 12 possible degrees, ($7°$ in $S(Ch)$ and 5 out of $S(Ch)$. For example let $Ch = Dm7$ and $S(Dm7) = (D, E, F, G, A, B, C)$ (dorian mode). Let $n = G$ then $Degree(G, S(Dm7)) = IV$. Let $n = Eb$ then $Degree(Eb, S(Dm7)) = bII$.

Notice that the musician's choice of $S(Ch)$ for a chord Ch is a crucial choice for the style of such a musician. Typically, it is possible to associate several scales to the same chord. For example, in Jazz music, a dominant chord is often substituted by the tritone chord. For example, let $Ch = G7$. Very traditional musicians could choose $S(G7) = (G, A, B, C, D, E, F)$ (mixolydian mode of Cmaj scale). An alternative choice is $S(Ch) = (Db, Eb, F, Gb, Ab, Bb, Cb)$ (mixolydian mode in Gbmaj scale). Obviously, the role of a note can be different depending on the chosen scale. In the previous example, let $n = F$. Then, if $S(G7) = (G, A, B, C, D, E, F)$ then $Degree(F, S(G7)) = VII$. Otherwise, if $S(Db7) = (Db, Eb, F, Gb, Ab, Bb, Cb)$ then $Degree(F, S(G7)) = III$. We can easily to say that the chord $G7$ has been substituted by the chord $Db7$. This process is known in Jazz as *substitution*. Further details can be found in [22].

3.2 Splicing Systems

Splicing systems are formal models for generating languages, i.e., sets of words [23]. We start with an initial set of words and we apply to these words the splicing operation by using rules in a given set. The set of generated words is joined to the initial set and the process is iterated on this new set until no new word is produced. The *language generated* is the collection of all these words.

Formally, a *splicing system* is a triple $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, where $\mathcal{A}$ is a finite alphabet, $\mathcal{I} \subseteq \mathcal{A}^*$ is the initial language and $\mathcal{R} \subseteq \mathcal{A}^*|\mathcal{A}^*\$\mathcal{A}^*|\mathcal{A}^*$ is the set of rules, where $|, \$ \not\in \mathcal{A}$. A splicing system $\mathcal{S}$ is finite when $\mathcal{I}$ and $\mathcal{R}$ are both finite sets. Let $L \subseteq \mathcal{A}^*$. We set $\gamma'(L) = \{w', w'' \in \mathcal{A}^* \mid (x, y) \vdash_r (w', w''),\ x, y \in L, r \in \mathcal{R}\}$. The definition of the splicing operation is extended to languages as follows: $\gamma^0(L) = L, \gamma^{i+1}(L) = \gamma^i(L) \cup \gamma'(\gamma^i(L)),\ i \geq 0$, and $\gamma^*(L) = \bigcup_{i \geq 0} \gamma^i(L)$.

Definition 1. *Let $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$ be a splicing system. We denote by $L(\mathcal{S}) = \gamma^*(\mathcal{I})$ the* splicing language generated *by $\mathcal{S}$. We say that L is* a splicing language *if there exists a splicing system $\mathcal{S}$ such that $L = L(\mathcal{S})$.*

The interested reader can refer to [24] for a detailed study of the theory of formal languages, and to [23,25] for further readings.

In [8] authors describe a music composer based on a splicing system for 4-voice chorale-like music. Similar to this earlier approach, the basic idea is to treat music compositions as words and to view the music compositional process as the result of operations on words.

3.3 One-Class Support Vector Machine (OCSVM)

A OCSVM algorithm maps input data into a high dimensional feature space (via a kernel) and iteratively finds the maximum margin hyperplane which best separates the training data from the origin. The OCSVM may be viewed as a regular two-class SVM where all the training data lies in the first class, and the origin is taken as the only member of the second class [26].

In terms of recognition, in this work we faced the following problem: given a melody, we want to verify whether it is coherent with a music performer's style. As we will see in Sect. 4, the idea is to associate a vector of significant features to the melody, and to use a One-class SVM for mapping the vector belonging to the training data (corpus of solos). Given the feature vector v_i corresponding to the improvised melody m_i, one-class SVM gives us the classifier (rule) $f(v_i) = \langle w, v_i \rangle + b$, where $\langle w, v_i \rangle + b$ is the equation of the hyperplane, with w being the vector normal to this hyperplane and b being the intercept. OCSVM solves an optimization problem of finding the rule f with the maximum geometric margin. We can use f to assign a label to a test example v_i. If $f(v_i) \geq 0$ then m_i is considered out of style, otherwise is considered in the style.

3.4 Music Composition Using LSTM Recurrent Neural Networks

We assume that the reader is familiar with the basics of machine learning more specifically artificial neural networks (ANN), but for further information see [27].

As well explained in [27], in the basic feedforward ANN there is a single direction in which the information flows: from input layer to output layer. But in a recurrent neural network (RNN), this direction constraint does not exist. The idea is to create an internal state (internal memory) of the network which allows it to exhibit dynamic temporal behavior.

The most straightforward way to compose music with an RNN is to use it as single-step predictor [28–30]. The network learns to predict notes at time $t + 1$ using notes at time t as inputs. A feedforward ANN would have no chance of composing music in this way, since it lacks the ability to store any information about the past. In "theory" an RNN does not suffer from this limitation. However, in "practice", RNNs do not perform very well this task given the problem of *vanishing gradients* which makes difficult for the networks to deal correctly with long-term dependencies. LSTM has succeed in similar domains where other RNNs have failed, and several works [31] showed that it is also a good mechanism for learning to compose music. For further information about LSTM see [32].

4 The Recognizer

Our intuition is that in each music perfomer it is possible to found music features that characterize his specific style. Thus, given a corpus of solos by several performers, to recognize a specific style we can extract such significant features and using them to make recognition. We remark that in our approach such features are *automatically* extracted from the corpus. In the following, we first introduce the model to extract the features and then present our machine learning approach. We will indicate with $\mathcal{M}$ the corpus solos by several musicians used for the definition of the recognizer. We assume to have a small subset $\mathcal{M'}_X$ that are melodies improvised by a specific musician X. So we have a classification problem in which the melodies in $\mathcal{M}$ can belong to two possible classes (the class of melodies coherent with the X's style and the class of melodies not coherent with the X's style. We remark that all the melodies in $\mathcal{M'}_X$ was improvised by X. Thus, the melodies in $\mathcal{M} - \mathcal{M'}_X$ can be coherent with the X's style or not. The goal of the classifier is to classify correctly the melodies in $\mathcal{M}$ in these two classes. Let v_j be the feature vector of some melody m_j, obtained through the feature extraction model to be discussed shortly. The recognizer will be defined to classify m_j by using v_j as input. In the following sections we indicate the recognizer with $\mathcal{R}$.

4.1 Feature Model

We are interested in a model that captures the most significant features of a melody. We use the n-gram model introduced in [33], which identifies *tokens* in melodies whose importance can than be determined through some *statistical measure*. An n-gram refers to n tokens which are dependent on each other.

Once n-grams have been constructed by the training set of melodies, we use a statistical measure to calculate their relative importance. The result is list of n-grams, ordered according to their importance. Depending on the value of n we can have different models, and different size of such a list, so for practical reasons, we fix a maximum size of the list. Notice that the n-grams in such a list have been selected according to their importance, but there is no theoretical support to their style classification capability. Thus, we apply a Random Forest selection procedure to keep those that better contribute to the classification (details in Sect. 7). In Fig. 1 we provide an overall view of the features extraction process.

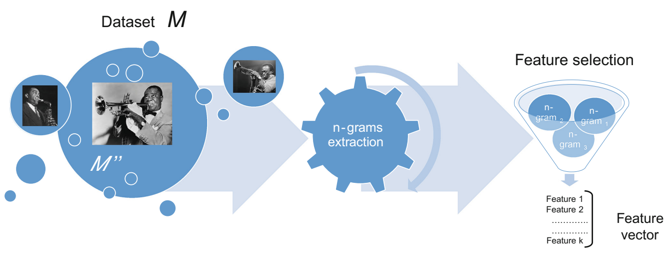

Fig. 1. Feature extraction process.

As we will see in Sect. 7, for each experiment we have tried several values of n but found no significant improvement beyond $n = 24$. In the following paragraphs, we will describe the tokens and statistical measure used in our approach.

The token. In our approach a token is the set of relevant information about a music note. Thus a n-gram represents a sequence of n notes. In order to describe a music note we decided to use three parameters (k_1, k_2, and k_3) that better explain the relation between melody and harmony, and specifically the role of the note respect to the chord in which is played. We refer to such parameters as *token's parameters*. The parameter k_1 indicates the *name chord*. In our approach we consider 12 possible music name chords, i.e., C, $C\#$, D, $D\#$, E, F, $F\#$, G, $G\#$, A, $A\#$, B. Thus we have 12 possible values for k_1. Specifically 0 for C, 1 for $C\#$, 2 for D and so on. The parameter k_2 refers to the *type chord*. In order to consider the most used type of chords, in this work we consider the chord types derived from the modes of the major, melodic minor and harmonic minor scales (see [22] for further details). In Table 1 we describe such type chords.

Table 1. Modes of the major, melodic minor and harmonic minor scales.

Description $S(Ch)$	Mode	Chord type	k_2 value
Major scale			
ionic	I	maj7	0
dorian	ii	m7	1
phrygian	iii	m7b9	2
lydian	IV	maj7#11	3
mixolydian	V	7	4
aeolian	vi	m7b6	5
locrian	vii	m7b5	6
Melodic minor scale			
ipoionic	i	m(maj7)	7
dorian b2	ii	m7b9	8
augmented lydian	III	maj7#5	9
dominant lydian	IV	7#11	10
mixolydian b6	V	7b6	11
locrian #2	vi	m7b5#2	12
superlocrian	vii	7alt	13
Harmonic minor scale			
ipoionic b6	I	m(maj7)	14
locrian #6	ii	m7b5b9b13	15
augmented lydian	iii	maj7#5	16
minor lydian	iv	m7#11	17
mixolydian b2b6	V	7b9b13	18
lydian #2	VI	maj7#9#11	19
diminished superlocrian	vii	$\circ$	20

The parameter k_3 indicates the *role* of the note respect to the chord. As explained in Sect. 1 for each scale there are 12 possible positions and so k_3 is an integer in the range from 0 to 11.

To summarize, the *3-tuple* $K^i = [k_1^i, k_2^i, k_3^i]$ is the token that describes the music note played at i^{th} time interval. For example $K^6 = [9, 4, 7]$ says that at the 6^{th} time interval the music note played has degree V ($k_3 = 7$ means degree V) in the scale corresponding to the chord $Bb7$ ($k_1 = 9$ means name chord Bb and $k_2 = 4$ means type chord 7). Thus the note played is D.

We remark that in our approach, the duration of the note is not considered as parameter of the token. Consequently, the rhythmic structure of melodies is not considered in this phase. This because in our opinion, the duration information is not important for the role respect to the chord, but it is important for the performance and the execution of the melody. Thus, as we will see in Sect. 6.5 the duration of the notes will be established by the composer at the end of the composition process, by using a specific operator.

The statistical measure. In our approach the statistical measure used to evaluate the relative importance of n-grams is the *term frequency with inverse document frequency* ($tfidf$). Central to this measure is the term t. In our approach a term t is a n-gram, i.e., a sequence of n tokens, which corresponds to a sequence of n music notes. We use the boolean term frequency (tf) measure such that $tf(t, m_j) = 1$ if $t \in m_j$ (the sequence of notes corresponding to t occurs in the melody m_j), 0 otherwise. The inverse document frequency measure *idf* is defined as: $idf(t, m_j, \mathcal{M}) = \log(|\mathcal{M}|/|\{\ m_j \in \mathcal{M} : t \in m_j\ \}|)$.

Finally we obtain tf with *idf* for each $t \in m_j$ over the corpus $\mathcal{M}$, as: $tfidf(t, m_j, \mathcal{M}) = tf(t, m_j) \cdot idf(t, m_j, \mathcal{M})$.

Intuitively the $tfidf$ measure gives more weight to terms that are less common in $\mathcal{M}$, since such terms are more likely to make the corresponding melody stand out. The $tfidf$ measure thus transform our corpus $\mathcal{M}$ to the feature vector space. The feature vector corresponding to a melody m_j is denoted by v_j. The ith component of v_j, denoted with $v_j[i]$ is equal to $tfidf(t_i, m_j, \mathcal{M})$.

4.2 The OCSVM classifier

As explained before, let m_j be a melody we define a feature vector corresponding to m_j denoted by v_j. Such a vector contains an element for each significant feature of the melody, and the value of this element is the $tfidf$ explained in Sect. 4.1. In our approach we use a OCSVM for mapping v_j. The OCSVM gives us the classifier (rule) $f(v_j) = \langle w, v_j \rangle + b$, where $\langle w, v_j \rangle + b$ is the equation of the hyperplane, with w being the vector normal to this hyperplane and b being the intercept. If $f(v_j) \geq 0$ then m_j is considered as melody not coherent with the X's style, otherwise m_j is coherent with the X's style. As we will see in Sect. 7 we use the traditional supervised two-class support vector machine (SSVM) [34] as benchmark for the performance of our OCSVM classifiers. In Fig. 2 we provide an overall view of the recognition process.

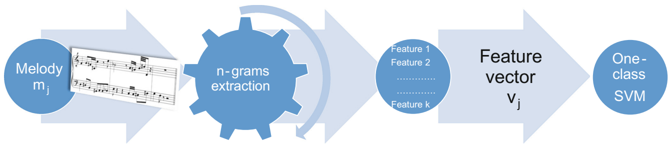

Fig. 2. Recognition process.

5 The Predictor

In this section we describe a LSTM network to predict musical patterns that follows the style learned by the recognizer. In our framework, we first build the recognizer $\mathcal{R}$ as described in Sect. 4, and then we build the predictor by using information and data from $\mathcal{R}$.

Thus, let $\mathcal{M}$ be a corpus of solos and $\mathcal{M}'_X \subseteq \mathcal{M}$ be a set of solos performed by a musician X. Let $\mathcal{R}$ be the recognizer and n the value used for the construction of the n-grams as described in Sect. 4.1. Then, the idea is to define a machine learning predictor $\mathcal{P}$ that given a n-gram at time i have to predict the n-gram at time $t+1$. This equivalent to saying that given the sequence of n music notes at time i, $\mathcal{P}$ has to predict the sequence of n music notes at time $i+1$.

The training set. We define a data set of n-grams as follows. Let $calT \subseteq \mathcal{M}$ be the training set used for the training of $\mathcal{R}$. For each $m_j \in \mathcal{T}$ such that $\mathcal{R}$ says that m_j is coherent with the X's style ($f(m_j) < 0$), we consider the sequence of n-grams extract by m_j. So, let $Ngrams(m_j) = (ng_1, \ldots, ng_{k_j})$ be the sequence of n-grams extract from m_j, we insert the pair (n_i, n_{i+1}) in the training set for $\mathcal{P}$, for each $1 \leq i \leq k_j - 1$.

The architecture. In our approach the predictor $\mathcal{P}$ is a LSTM network and we now describe the steps and reasons for its definition.

1. First, we define a ANN that must predict the music note at time $i+1$ given music note at time i. As described in Sect. 4, each note is a token K^i which is a triple $K^i = [k_1^i, k_2^i, k_3^i]$. Thus, input and output layers have size 3. In our approach we use four hidden layers having size 3.
2. To add "recurrency", we take the output of each hidden layer, and feed it back to itself as an additional input. Each node of the hidden layer receives both the list of inputs from the previous layer and the list of outputs of the current layer in the last time.
3. To solve the problem of shot-term memory we use LSTM nodes.
4. We need that the network has to be *(1) time-invariant*, i.e., identical for each step, and *(2) note-invariant*, i.e., identical for each note. To this, we build a stack of n identical RNN networks, one for each token. The overall network can be viewed as a predictor of the n-gram at time $i+1$ given n-gram at time i. We use a "biaxial RNN" approach: there are two axis, i.e., time-axis and note-axis; each recurrent layer transforms input to outputs, and also sends recurrent connections along one of these axis. The first two layers

have connections across time steps, but are independent across notes. The last two layers have connections between notes, but are independent between time steps.

In Fig. 3 we provide an overall view of the construction of $\mathcal{P}$. As we will see in Sect. 6, it has a fundamental role for the construction of the composer.

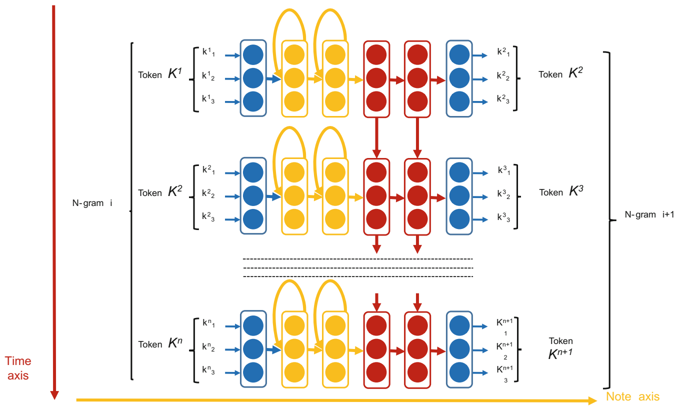

Fig. 3. LSTM recurrent neural network architecture.

6 The Composer

In this section we describe a music splicing system to compose melodies coherent with the style learned by the recognizer. In our framework, we first build the recognizer $\mathcal{R}$ as described in Sect. 4, then we build the predictor $\mathcal{P}$ as described in Sect. 5 and finally we build the composer.

Thus, let $\mathcal{M}$ a corpus of transcribed solos and $\mathcal{M'}_X \subseteq \mathcal{M}$ a corpus of solos performed by a specific musician X. The idea is to build a music splicing composer which produces melodies coherent with the X's style.

As explained in [8] in order to define a music splicing composer we need to define an alphabet, an initial set and a set of rules. Such a system is defined by an initial set of melodies coherent with the X's style, and a set of rules built by using $\mathcal{P}$. The language generated contains words that represent pieces of "new" melodies coherent the X's style. Formally, a music splicing system is a triple $\mathcal{S}_{\text{MSS}} = (\mathcal{A}_{\text{MSS}}, \mathcal{I}_{\text{MSS}}, \mathcal{R}_{\text{MSS}})$. Among the generated words we choose the best solution according to the evaluation function that we will define in Sect. 6.4.

6.1 The Alphabet $\mathcal{A}_{\text{mss}}$

We set $\mathcal{A}_{\text{MSS}} = \mathcal{A}_N \cup \mathcal{A}_T \cup \mathcal{A}_D \cup \mathcal{A}_S$ where $\mathcal{A}_N$ is the *chord name alphabet*, $\mathcal{A}_T$ is the *chord type alphabet*, $\mathcal{A}_D$ is *note degree alphabet* and $\mathcal{A}_S$ is the *separator alphabet.* Specifically, $\mathcal{A}_N = \mathcal{A}_D = \{0, \ldots, 11\}$, $\mathcal{A}_T = \{0, \ldots, 20\}$ and $\mathcal{A}_S = \{\tau, \mu_N, \mu_T, \mu_D\}$. We use $\mathcal{A}_{\text{MSS}}$ to represent melodies as words. As explained in Sect. 4 each token represents a music note. Thus, given a melody $m = (n_1, \ldots, n_l)$, for each note n_i, the token K^i is represented as a word w_i over $\mathcal{A}_{\text{MSS}}$. Specifically $w_i = \mu_N x_i \mu_T y_i \mu_D v_i$, where $x_i \in \mathcal{A}_N$, $y_i \in \mathcal{A}_T$ and $y_i \in \mathcal{A}_D$, for each $1 \leq i \leq l$. Thus, m is represented by the word $\mathcal{W}(m) = \tau w_1 \tau w_2 \tau \cdots \tau w_n \tau$.

Example 1. Let us consider the melody m shown in Fig. 4 (first 3 measures of the standard Jazz "All blues" of Miles Davis). As explained before, our representation does not take into account the duration of the notes (and tied notes are considered as a single note). We have $m = (n_1, \ldots, n_{10})$. Each n_i is played in G7 chord. The chord name G has value 7. Chord G7 is the associated scale is the mixolydian mode of the major scale (see Table 1), thus the chord type has value 4. Now, let us analyze the role of notes respect the G7. We have $n_1 = n_3 = n_5 = n_7 = n_9 = B$ which has degree iii (position value 4), $n_2 = n_6 = n_{10} = D$ which has degree V (position value 7) and $n_4 = n_8 = C$ has degree IV (position value 5). Thus, we have 10 tokens: $K^1 = K^3 = K^5 = K^7 = K^9 = (7, 4, 4)$, $K^2 = K^6 = K^{10} = (7, 4, 7)$ and $K^4 = K^8 = (7, 4, 5)$. So, $\mathcal{W}(m) = \tau w_1 \tau \cdots \tau w_{10} \tau$ and $w_1 = \mu_N 7 \mu_T 4 \mu_D 4$, $w_2 = \mu_N 7 \mu_T 4 \mu_D 7$, $w_3 = \mu_N 7 \mu_T 4 \mu_D 4$, $w_4 = \mu_N 7 \mu_T 4 \mu_D 5$, $w_5 = \mu_N 7 \mu_T 4 \mu_D 4$, $w_6 = \mu_N 7 \mu_T 4 \mu_D 7$, $w_7 = \mu_N 7 \mu_T 4 \mu_D 4$, $w_8 = \mu_N 7 \mu_T 4 \mu_D 5$, $w_9 = \mu_N 7 \mu_T 4 \mu_D 4$ and $w_{10} = \mu_N 7 \mu_T 4 \mu_D 7$.

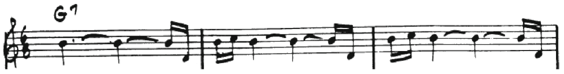

Fig. 4. First 3 measures of Miles Davis's All Blues.

6.2 Initial Set and Rules Definition

The idea is to start from an initial set of melodies that we known to be coherent with the X's style. Thus, we consider the following set of n-grams (the n used by $\mathcal{R}$): let $\mathcal{T}$ be the training set used for the training of $\mathcal{R}$. For each $m_j \in \mathcal{T}$ such that $\mathcal{R}$ says that m_j is coherent with the X's style ($f(m_j) < 0$) we consider the list $Ngrams(m_j)$ of n-grams extract by m_j. So, let $Ngrams(m_j) = (ng_1, \ldots, ng_{k_j})$, we insert $\mathcal{W}(ng_i)$ in the $\mathcal{I}_{\text{MSS}}$, for each $1 \leq i \leq k_j - 1$.

We remark that the definition of the rules is crucial because they determine the language being generated. Unlike the problem described in [8], in which it was possible to model rules by using the theory of music harmony, in our case we have not a method to model rules to compose melodies coherent with the X's style. Thus, we can use $\mathcal{P}$ as follows.

1. *Group 1 (forcing sequence n-grams).* Let $w_i \in \mathcal{I}_{\text{MSS}}$ and n_i such that $w_i = \mathcal{W}(n_i)$, these rules force the composer to paste words corresponding to n-grams that really follows n_i in the training set $\mathcal{T}$. Formally, let $w_i \in \mathcal{I}_{\text{MSS}}$, and n_i such that $w_i = \mathcal{W}(n_i)$, we define the rule $r = w_i|\epsilon\$\epsilon|\mathcal{W}(\mathcal{P}(n_i))$ where, $\mathcal{P}(n_i)$ is the n-gram predicted by $\mathcal{P}$ with n_i as input.
2. *Group 2 (forcing sequence tokens).* Given two words $w_i, w_j \in \mathcal{I}_{\text{MSS}}$ and n_i, n_j such that $w_i = \mathcal{W}(n_i)$, and $w_j = \mathcal{W}(n_j)$. If n_i and n_j share a token, then these rules force the composer to cut w_i and w_j in correspondence of such a token, and paste the remaining words. Formally, let $w_i, w_j \in \mathcal{I}_{\text{MSS}}$, n_i such that $w_i = \mathcal{W}(n_i)$, n_j such that $w_j = \mathcal{W}(n_j)$, where $n_i = (k_{i,1}, \ldots, k_{i,n})$, $n_j = (k_{j,1}, \ldots, k_{j,n})$ and there exist l_1, l_2 such that $k_{i,l_1} = k_{j,l_2}$. Then, we define the rule $r = \mathcal{W}(k_{i,1}, \ldots, k_{i,l_1-1})\epsilon\$\epsilon|\mathcal{W}(k_{j,l_2}, \ldots, k_{j,n_j})$.

6.3 Implementation Details

In general, given a splicing system $\mathcal{S} = (\mathcal{A}, \mathcal{I}, \mathcal{R})$, the generated language $L(\mathcal{S})$ is an infinite set of words, and the number of iterations of the splicing operation to generate it is unbounded. Of course, for practical reasons, we need to fix bounds for both these parameters (cardinality of the language and number of iterations). Thus, we fix a number k of iterations and a maximal cardinality p_{max}. We also define k languages as follows. We set $L_0 = \mathcal{I}_{\text{MSS}} = \gamma^0(\mathcal{I}_{\text{MSS}})$. For any i, $1 \leq i \leq k$, we consider $L'_i = L_{i-1} \cup \gamma'(L_{i-1})$, which corresponds to enlarge L_{i-1} by an application of all the rules in $\mathcal{R}_{\text{MSS}}$ to all possible pairs of words in L_{i-1}. If $\text{Card}(L'_i) \leq p_{max}$, then $L_i = L'_i$. Otherwise, L_i is obtained from L'_i by erasing the $\text{Card}(L'_i) - p_{max}$ words in L'_i that are the worst with respect to an evaluation function, i.e., the quality of the melody in terms of stylistic coherence. Therefore, to measure the quality of the compositions and to choose the better solutions we consider such a function. Finally, we define $L(k, p_{max}) = \cup_{1 \leq i \leq k} L_i$ as the (k, p_{max})-language generated by $\mathcal{S}_{\text{MSS}}$. We remark that $L(k, p_{max})$ is the language considered during the experiments described in Sect. 7.

6.4 The Evaluation Function

As we will see in Sect. 7, in order to use the composer, from a practical point of view we need to set some parameters, such as, maximum size of the generated language and so a function f_e to evaluate the melodies produced, used to select at each splicing step the best compositions.

As for the definition of the rules described before, also in this case we have not music rules to use for defining a function that evaluate a composition in terms of "stylistic" goodness. So the idea is to use the predictor $\mathcal{P}$ as follows. Let w be a word generated by the composer $\mathcal{S}_{\text{MSS}}$. Let $m = (n_1, \ldots, n_l)$ the melody such that $w = \mathcal{W}(m)$. Now, let $Ngrams(m) = (ng_1, \ldots, ng_{l-1})$ be the sequence of n-grams extract from m. We define f_e as:

$$f_e(m) = \sum_{1 \leq i \leq l-1} (tfidf(ng_i, m, \mathcal{M})) + Diff(ng_{i+1}, \mathcal{P}(ng_i))$$

where $Diff(ng_i, \mathcal{P}(ng_i))$ is the *difference* between ng_{i+1} and $\mathcal{P}(ng_i)$ that is the n-gram predicted by $\mathcal{P}$ with ng_i as input. Such a difference is defined as follows: let $K^{i+1} = [k_1^{i+1}, k_2^{i+1}, k_3^{i+1}]$ be the token for ng_{i+1} and $K'^{i+1} = [k_1'^{i+1}, k_2'^{i+1}, k_3'^{i+1}]$ be the token for $\mathcal{P}(ng_i)$. Then $Diff(ng_{i+1}, \mathcal{P}(ng_i)) = |k_1^{i+1} - k_1'^{i+1}| + |k_2^{i+1} - k_2'^{i+1}| + |k_3^{i+1} - k_3'^{i+1}|$.

6.5 The Rhythmic Transformation of the Melodies

We remark that when a word w is generated, it only represents a sequence of music notes $m = (n_1, \ldots, n_k)$. In order to add a rhythmic structure to m we use an operator that applies the following operations (each operation is performed with a uniform distribution of probabilities):

1. *Duration:* initially the duration of each note is assumed to be equal a beat duration. For each note n_i, the operator changes the duration of $n_i \in m$. If the duration of n_i is increased of a value inc then the duration of the remaining notes is decreased of $inc/(k-1)$; otherwise, if the duration of n_i is decreased of a value dec then the duration of the remaining notes is increased of $dec/(k-1)$.
2. *Rest notes:* $(n_1, \ldots, n_k)$ for each note $n_i \in m$, the operator inserts a rest note having the duration dur_i of n_i. So the duration of the remaining notes is decreased of dur_i/k.
3. *Tied notes:* For each note $n_i, n_{i+1} \in m$ such that $n_i = n_{i+1}$, the operator inserts a ligature between n_i and n_{i+1}.
4. *Triplet notes:* Given the list of notes $(n_1, \ldots, n_k)$, for each note n_i, n_{i+1}, n_{i+2}, the operator creates a triplet by using these notes.

7 Experimental Analysis

In this section we report the results of tests that we carried out to assess the validity of our approach. We implemented the system in Python by using the Anaconda and Scikit-learn libraries. We focused on Jazz music and thus we create a data set $\mathcal{M}$ of transcribed solos in MusicXML format, from three very popular Jazz musicians: Louis Armstrong (A), Charlie Parker (P), and Miles Davis (D). Formally, $\mathcal{M} = \mathcal{M}_\mathcal{A} \cup \mathcal{M}_\mathcal{P} \cup \mathcal{M}_\mathcal{D}$ where $\mathcal{M}_\mathcal{A}$, $\mathcal{M}_\mathcal{P}$ and $\mathcal{M}_\mathcal{D}$ are the sets of Armstrong's, Parkers's and Davis's solos respectively. In details, $|\mathcal{M}_\mathcal{A}| = 50$, $|\mathcal{M}_\mathcal{P}| = 50$ and $|\mathcal{M}_\mathcal{D}| = 50$, so $|\mathcal{M}| = |\mathcal{M}_\mathcal{A}| + |\mathcal{M}_\mathcal{P}| + |\mathcal{M}_\mathcal{D}| = 150$. The average number of notes for each melody in $\mathcal{M}$ is 318.

To validate our approach we performed 3 experiments, one for each chosen musician. For each experiment we fixed $X \in \{A, P, D\}$ and we selected the reference set $\mathcal{M}' \subseteq \mathcal{M}_\mathcal{X}$. Then, by using $\mathcal{M}_\mathcal{X}$ and $\mathcal{M}'$ we build a OCSVM recognizer $\mathcal{R}_X$ (Sect. 4), a predictor $\mathcal{P}_X$ (Sect. 5) and a composer $\mathcal{C}_X$ (Sect. 6). The aim of each experiment is to validate and to verify the capability of: *(1)* $\mathcal{R}_X$ in recognizing the X's style, *(2)* $\mathcal{P}_X$ in predicting patterns that follows the X's style and *(3)* $\mathcal{C}_X$ in composing melodies that follows the X's style.

Recognition performances. First we use a traditional supervised two-class support vector (SSVM) machine as a benchmark for the performance of $\mathcal{R}_X$. For $\mathcal{R}_X$ we build a training set by considering the 80% of melodies in $\mathcal{M}'$ (randomly chosen). For the SSVM, we build a training set by considering the 80% of melodies in $\mathcal{M}_{\mathcal{X}}$ (randomly chosen) and a testing set by considering the 20% of melodies in $\mathcal{M}_{\mathcal{X}}$ (randomly chosen). As we can see in Table 2 (left) best results have been obtained with $n = 24$, and in this case two classifiers achieve very similar rates, with true positive and negative rates of up to 0.97 and false positive and negative of only 0.03. This prove that the performance of $\mathcal{R}_X$ are comparable to the traditional SSVM. $\mathcal{R}_X$, where training set of only a subset of melodies in $\mathcal{M}_{\mathcal{X}}$ is used for the learning, has the advantage that it requires fewer examples of melodies that follows the X's style. As a result we obtain that the one-class support vector machines were proven to roughly as good as two-class support vector machines for this problem.

Prediction efficacy. In this section we describe the results of a study aiming at assess the efficacy of $\mathcal{P}$ in terms of abilities in predicting patterns that follows the X's style. As explained in Sect. 5 the accuracy of $\mathcal{P}$ is also validated and compared against a feedforward ANN and a recurrent ANN trained on the same training set. Table 2 (right) summarizes the (most significant) best average results (over the three experiments) about the prediction rate. As we can see, in a range of 22000 epochs, we have obtained the highest prediction rate using the LSTM recurrent neural network, with Back propagation training, $M = 0.6$ (momentum) and $L = 0.7$ (learning rate).

Table 2. Classifiers performances (left) and Test with average prediction rate (right)

Feature model	Classifier	$\mathcal{M}_X$		$\mathcal{M} - \mathcal{M}_X$	
		Yes	Not	Yes	Not
$n = 12$	$\mathcal{R}_X$	0.91	0.09	0.06	0.94
	SSVM	0.90	0.1	0.04	0.96
$n = 16$	$\mathcal{R}_X$	0.94	0.06	0.05	0.95
	SSVM	0.94	0.06	0.04	0.96
$n = 24$	$\mathcal{R}_X$	0.3	0.97	0.04	0.96
	SSVM	0.3	0.97	0.03	0.97

Representation	M	L	Prediction rate [%]
LSTM	0.6	0.7	95.1
LSTM	0.5	0.4	93.9
Recurrent NN	0.5	0.5	89.1
ANN	0.4	0.6	85.4
LSTM	0.7	0.3	85.1

Music quality. The quality and the stylistic coherence of music produced by $\mathcal{C}_X$ has been evaluated by music experts among conservatory's teachers and professional musicians. All participants had more than 10 years of experience in the music field. For each experiment we select the best 4 solutions (available online[1]) produced by $\mathcal{C}_X$ according to the evaluation function f_e (see Sect. 6.4). We asked participants to listen to such a list and respond to the following questions: *(1) "How do you rate the quality of music?"*, and *(2) "How do you rate the*

[1] http://goo.gl/FWn2EX.

coherence of the music with the X's style?" (rating on a 7-point Likert scale). We also asked participants to provide a motivation about their judgments.

Our experts rated very positively both quality and stylistic coherence (M = 6.2 and M = 6.5, respectively). They also were impressed about the soundness of the music produced by $\mathcal{C}_X$. Nevertheless, some participants felt that in some case the rhythm is too elaborate. The motivation is that the automatic rhythmic operator defined in Sect. 6.5 sometimes is not able to create appropriate rhythms.

8 Conclusion

Several works present different systems for recognizing a music style, mostly based on machine learning approach. Conversely, relatively few works describe automatic systems composer of music coherent with a specific style.

In this paper we propose a new approach for the recognition of a music performer's style and automatic composition of melodies coherent with such a style. Specifically, we describe a machine learning approach for recognition based on a one-class machine learning classifier, and a bio-inspired approach for automatic composition based on music splicing systems. We performed several tests to analyze the efficacy of our system in terms of a style recognition and coherence of the produced melodies with such a style. Performances of our classifier are comparable with the traditional two-class SVM and it is able to achieve an accuracy of 97%. Some future works will include new experiments to widen the corpus of solos to consider other Jazz musicians and other musical genres. Furthermore, to address comments obtained during the music quality evaluation we will define a stylistic rhythmic operator, by using the same approach used for the melodic features. The idea is to individuate the more significant rhythmic features of a music style, and using such features during the composition process.

References

1. Pampalk, E., Dixon, S., Widmer, G.: Exploring music collections by browsing different views. In: Proceedings of the Fourth International Conference on Music Information Retrieval, Baltimore (2003)
2. Whitman, B., Flake, G., Lawrence, S.: Artist detection in music with minnowmatch. In: Proceedings of the 2001 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing XI, pp. 559–568. IEEE (2001)
3. Soltau, H., Schultz, T., Westphal, M., Waibel, A.: Recognition of music types. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (1998)
4. Pachet, F., Westermann, G., Laigre, D.: Musical data mining for electronic music distribution. In: First International Conference on WEB Delivering of Music (WEDELMUSIC 2001), Florence, Italy, November 23–24, 2001, pp. 101–106 (2001)
5. Dannenberg, R.B., Thom, B., Watson, D.: A machine learning approach to musical style recognition. In: International Computer Music Conference, pp. 344–347 (1997)

6. Tzanetakis, G., Ermolinskyi, A., Cook, P.: Pitch histograms in audio and symbolic music information retrieval. In: Fingerhut, M. (ed.) Third International Conference on Music Information Retrieval: ISMIR 2002, pp. 31–38 (2002)
7. Miranda, E.: Composing Music with Computers. Focal Press, Oxford (2001)
8. Felice, C., Prisco, R., Malandrino, D., Zaccagnino, G., Zaccagnino, R., Zizza, R.: Chorale music splicing system: an algorithmic music composer inspired by molecular splicing. In: Johnson, C., Carballal, A., Correia, J. (eds.) EvoMUSART 2015. LNCS, vol. 9027, pp. 50–61. Springer, Cham (2015). doi:10.1007/978-3-319-16498-4_5
9. Ebcioglu, K.: An expert system for harmonizing four-part chorales. In: Machine Models of Music, pp. 385–401 (1992)
10. Sundberg, J., Askenfelt, A., Frydén, L.: Musical performance: a synthesis-by-rule approach. Comput. Music J. **7**(1), 37–43 (1983)
11. Friberg, A.: Generative rules for music performance: a formal description of a rule system. Comput. Music J. **15**(2), 56–71 (1991)
12. Cope, D.: Experiments in Musical Intelligence. Computer Music and Digital Audio Series, A-R Editions (1996)
13. Lehmann, D.: Harmonizing melodies in real-time: the connectionist approach. In: Proceedings of the International Computer Music Association, pp. 27–31 (1997)
14. Wiggins, G., Papadopoulos, G., Amnuaisuk, S., Tuson, A.: Evolutionary methods for musical composition. In: CASYS 1998 (1998)
15. Horner, A., Goldberg, D.: Genetic algorithms and computer assisted music composition. Technical report, University of Illinois (1991)
16. Biles, J.A.: GenJam: a genetic algorithm for generating jazz solos. In: International Computer Music Conference, pp. 131–137 (1994)
17. Horner, A., Ayers, L.: Harmonization of musical progression with genetic algorithms. In: International Computer Music Conference, pp. 483–484 (1995)
18. Biles, J.A.: GenJam in perspective: a tentative taxonomy for GA music and art systems. Leonardo **36**(1), 43–45 (2003)
19. Prisco, R., Zaccagnino, R.: An evolutionary music composer algorithm for bass harmonization. In: Giacobini, M., Brabazon, A., Cagnoni, S., Caro, G.A., Ekárt, A., Esparcia-Alcázar, A.I., Farooq, M., Fink, A., Machado, P. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 567–572. Springer, Heidelberg (2009). doi:10.1007/978-3-642-01129-0_63
20. De Prisco, R., Zaccagnino, G., Zaccagnino, R.: Evobasscomposer: a multi-objective genetic algorithm for 4-voice compositions. In: Genetic and Evolutionary Computation Conference, GECCO 2010, Portland, Oregon, USA, July 7–11, pp. 817–818 (2010)
21. De Felice, C., De Prisco, R., Malandrino, D., Zaccagnino, G., Zaccagnino, R., Zizza, R.: Splicing music composition. Inf. Sci. **385—-386**, 196–212 (2017)
22. Levine, M.: The jazz theory book. Curci (2009)
23. Head, T.: Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. Bull. Math. Biol. **49**, 737–759 (1987)
24. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Addison-Wesley, Reading (2006)
25. Păun, G.: On the splicing operation. Discrete Appl. Math. **70**, 57–79 (1996)
26. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)
27. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus (2006)

28. Todd, P.M.: A connectionist approach to algorithmic composition. In: Todd, P.M., Loy, D.G. (eds.) Music and Connectionism, pp. 173–194. MIT Press/Bradford Books, Cambridge (1991)
29. Bharucha, J.J., Todd, P.M.: Modeling the perception of tonal structure with neural nets. Comput. Music J. **13**(4), 44–53 (1989)
30. Mozer, M.: Neural network music composition by prediction: exploring the benefits of psychoacoustic constraints and multi-scale processing. In: Connection Science, pp. 247–280 (1994)
31. Eck, D., Schmidhuber, J.: A First Look at Music Composition Using LSTM Recurrent Neural Networks. Technical report (2002)
32. Gers, F.A., Schmidhuber, J.: Recurrent nets that time and count. In: Proceedings of the IJCNN 2000, International Joint Conference on Neural Networks, Como, Italy (2000)
33. Hsiao, C.H., Cafarella, M., Narayanasamy, S.: Using web corpus statistics for program analysis. In: OOPSLA. ACM (2014)
34. Muller, K.R., Mika, S., Rt, G., Tsuda, K., Schlkopf, B.: An introduction to kernel-based learning algorithms. IEEE Trans. Neural Networks **12**(2), 181–201 (2001)