LickNet: Software Tools Collection for Guitar Licks Networks

Matteo Martelli

University of Bologna
matteo.martelli9@studio.unibo.it

Abstract. The purpose of the project described in this document is to provide a method to analyse the relations between musical note patterns in guitar solos. Given the large amount of guitar solo data available in the Web, it has been possible to develop an algorithm able to generate a network of guitar solos. An empirical study has then been conducted in order to identify some well known characteristics of the generated networks, which showed that it seems to satisfy the small world property[8]. Moreover, with the intent to provide an application for the generated networks, two interesting tools have been developed: a Licks Classifier, useful to identify to which artist a set of guitar licks belongs to, and a Licks Generator, useful to generate new licks which are intended to reflect a particular guitarist's style. Thanks to the intuitiveness and surprising effectiveness of implementing such relatively simple algorithms, the initial expectations about the project described in this document, have been satisfied.

1 Introduction

Musicians are often fascinated by how other musicians compose or improvise their music. A musician may also often feel some kind of relation between his music and some other musician's music. Even non expert music enthusiasts often wonder what makes a musician so unique and easily recognisable among many, or what makes a musician so similar to another. This frequently happens with guitarists for example, when people recognize a *Santana* guitar solo or a *Hendrix* guitar solo just by listening to a few notes, or when people say that *Steve Vai* sounds similar to *Joe Satriani*.[1]. One may also think that this is due to the fact that there are repetitions among some note patterns in a musician's repertoire or that similar note patterns emerge from their common musical influence or musical tastes.

Thus the idea for *LickNet* emerged from my effort to analyse the relations (modeled as networks) between note patterns in guitar solos (licks) as described in this document. The choice to focus on guitar solos has been made because of

[1] *Last.fm*, a collaborative website, assigns a "super" level of similarity between Steve Vai and Joe Satriani. This may very well be due to the fact that Steve Vai was his student.

the large amount of data available online but my work may easily be extended to other similar kinds of data. For example drum patterns, whole compositions, improvisations or even jam sessions may be considered. It has been chosen to model the data elements and their relations within a network as the analytical methods in graph theory are very useful.
The area of scientific research that is usually interested in the empirical examination of real-world networks is the study of *Complex Networks*. Different approaches and mathematical methods pertaining to the Complex Networks field of study have been considered and will be discussed in the next sections; including graph theory and small-world network comparison[1]
As said before, in addition to the network analysis tools, LickNet provides other features like the *Lick Classifier* and the *Lick Generator*. These functionalities are still in the early stages of development but they can be a starting point for some practical applications; e.g. to simply understand and replicate some guitarist's playing style or to create a full-fledged artificial agent able to "play" music and "interact" with human players.

2 Related Works

Several empirical studies about music have already been made using network analysis. In [2], networks have been used to model the correlation among musical tastes of different listener. In addition, the same paper shows a technique to create a network of musical groups and artists which are connected by the similarity of their audience. A similar approach has been used in other remarkable works [3][4].
The authors of [5] introduced a method to create a network of human language words, linking them through a correlation parameter such as their relative distance in a sentence, choosing a value of 2 as the maximum distance for forming links. This means that two words can be linked together if in a sentence one word is the successor or the successor of the successor of the other word. They made this choice for two reasons:

1. Many co-occurrences of words take place at a distance of one, e.g. 'red flowers' (adjective-noun), 'stay here' (verb-adverb), 'getting dark' (verb-adjective), etc.
2. Many co-occurrences of words take place at a distance of two, e.g. 'hit the ball' (verb-object), 'Mary usually cries' (subject-verb), 'Live in Boston' (verb-noun), etc..

In the section 5 it will be pointed out that a similar graph creation procedure as been used in this work, but with the maximum distance for forming links of 1. This because it is not trivial to determine a multi-step correlation value between musical notes in a notes lick. Use of a maximum distance greater than 1 may be tested in future versions.

3 LickNet

As stated before, LickNet offers the possibility to study the relations among note patterns in guitar solos through the creation and analysis of guitar licks networks. It comprises software tools with which it is possible to operate on these networks. These operations are currently two, a classifier and a generator of guitar licks.
Before getting into the software components, the next section is going to focus on how data is collected, structured and processed.

4 Data Representation

As many guitarists may already know, there are plenty of websites that store guitar sheets and tablatures. One of them is `www.ultimate-guitar.com` that counts more than 800000 sheets, which are mostly written in the guitar tablature notation. Guitar players usually choose this notation for its ease of use [2] but all this is instrument-specific. Also, guitar tablature is not standardized and different publishers adopt different conventions.

This means that a guitar tablature can be understood only by guitar players and its conversion to the standard notation or formal interpretation may result wearing.

For example the semantic of a guitar tablature changes even if a guitar is not standard tuned.
Fortunately there are various computer programs available for writing tablature. One of the most frequently used is *Guitar Pro* and the *Ultimate Guitar* servers store a large amount of tablature files encoded in this particular format. Another interesting software is *TuxGuitar*, a free and open source tablature editor that also supports the ability to import and export Guitar Pro files[6]. Moreover its source code can be freely re-used and adapted to any other application. LickNet, in fact, includes the TuxGuitar library with the purpose of importing tablature files retrieved online.
This way tablature can be interpreted by software and a guitar solo can be represented as a sequence of notes, each having the following properties:

- **String**: at which string the note is played
- **Value**: at which fret the note is played
- **Effects**: a collection of effects such as bending, vibrato, harmonic, etc.
- **Duration**: the duration of the note, represented as a fraction.

This is how the TuxGuitar library represents the notes, but some adaptation and additional piece of information is needed for my purpose. It has been decided to represent a single note in LickNet as the following:

[2] Guitar tablature removes the requirement for the player to remember the association between each note and the corresponding finger position on the fretboard since they are directly represented in numbers (which fret) for each string.

- **Base Note**: the value of the musical note. It is a value in the range of [0,12] irrespective of the octave. The 0th note is assigned to C, the 11th to B, the last one, and the 12th is the rest.[3]
- **Octave**: the octave of the note. The note pitch p is given by the equation $p = o \times 12 + b$, where o is the octave and b is the base note.
- **Time**: it's a floating point value in the range [0,1] indicating the duration of the note. The dotted and double dotted notes are also considered, as are the triplets. For example a quaver dotted note would have a time value of $\frac{1}{8} + \frac{1}{16} = 0.1875$ and a semi-quaver triplet note would have the time value of $\frac{1}{16} \times \frac{2}{3} \simeq 0.042$.
- **Bend Distance**: if a note has a bending effect, the bending distance is the offset from the base note to the bended note. For example if a F note (base note = 5) is bended to G than the distance is 2, but if it is bended to F# (half step bending) then the distance is 1. It is possible that the bending effect is defined with more complex properties: it may be defined with the *bending and release* property or with the *bending, release and bending again* property. Thus it has been decided to calculate the bending distance has the average bending distance.
- **Node Key**: it is an identifier for a note in the graph. It is essentially a string where the values of the previous fields except the octave are concatenated. For instance a semi-quaver triplet F note with a bending to G would have the resulting node key: "05:0.042:b2". There may be a case in which the bending should not be considered, in this case the bending distance offset will be added to the base note resulting in: "07:0.042".

In the next section it will be shown how the data model, introduced here, is used in the network creation.

5 Networks Generator

The network is modeled as a directed multigraph[7] and it is generated by scanning the input music sheet file. Let G be a directed multigraph defined as an ordered 4-tuple (N, E, s, t) with

- N a set of nodes,
- E a set of edges,
- $s : E \to N$, assigning to each edge its source node,
- $t : E \to N$, assigning to each edge its target node.

Moreover a node $n \in N$ contains a musical note identified by its Node Key. An edge $e \in E$ from a node s to a node t exists if the musical note contained in the node t was a successor, in the input music sheet, of the musical note contained in the node s. Also, there may exist more than one edge from a node s and t, insofar as a same musical note can be reached by another note many times with different octaves jumps. Thus an edge from s to t indicates with which octave jump the note contained in s can reach t in a graph walk step. An example of a multigraph with its corresponding sheet music is shown in figure 1. The octave

[3] A rhythmic period of silence between tones.

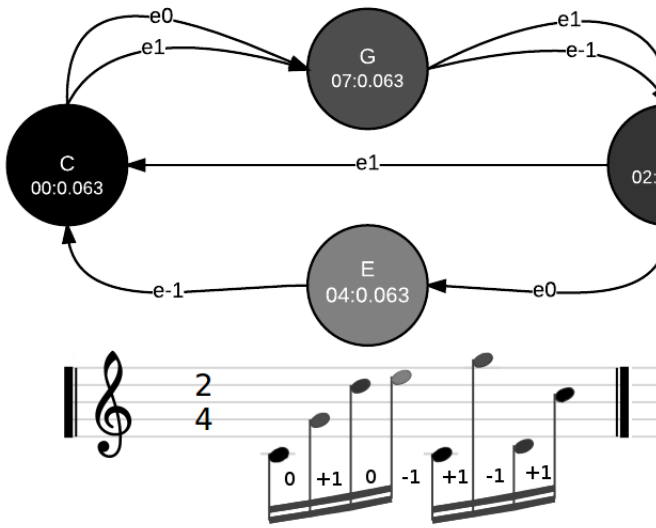

Fig. 1. A multigraph example. The corresponding input data used for the multigraph creation is represented with its sheet. The id of each edge indicates with which octave jump its source node can reach the target node.

informations are not included in the graph nodes because similar licks can be played at different octaves and otherwise it would be harder to underline the relations between them.
It has been decided to add a weight to each edge that corresponds to how many times an edge is passed through. This means that many walks from a node a to a node b with the same distance d but different total weight exist in the graph, and the walk more likely to correspond to a significant lick for the graph is the one with the highest total weight. A lick is intended to be significant if it is similar to the licks that are already used in the graph creation. The process that determines if a lick is significant for a certain graph will be covered in the next section.
A simplification of the model has been introduced in the development process: the directed multigraph is represented as a directed graph, where an edge is an object that contains an array which each i_{th} element indicates the pass-through frequency number for the octave corresponding to i. More specifically if $f(x)$ is the frequency number related to an octave jump x, then the i_{th} element of the array would be:

$$v(i) = f(i + N_o),$$

where N_o is the maximum octave increment or decrement possible. There are 5 octaves on a standard tuned guitar, thus with 6 octaves even the 7-strings guitars and most of the non-standard tuned guitars can be covered. Therefore the size of the array is 12 because a node s can reach in a single step a node t with an increment or decrement of at most 6 octaves. To clarify, if the node s

has reached with a single step the node t 5 times without any octave jump, 3 times with an octave jump of +1, 1 time with an octave jump of +2 and 2 times with an octave jump of -1, the resulting octave jumps array for the edge e that connects s and t would be:

$$v = (0, 0, 0, 0, 0, 2, 5, 3, 1, 0, 0, 0, 0)$$

A directed edge e from s to t exists if at least one element of the array is non-zero. The algorithm 1 shows how the graph creation process works.

Algorithm 1 Graph creation algorithm

```
function createGraph(sheets)
    G ← emptyGraph()
    for s in sheets do
        for n in s.notes do
            if pn then
                S_k ← genNodeKey(pn)
                T_k ← genNodeKey(n)
                if S_k ∉ G.nodes then
                    createNode(G, S_k)
                end if
                if T_k ∉ G.nodes then
                    createNode(G, T_k)
                end if
                E_k ← A_k + B_k                          ▷ Strings concatenation
                if E_k ∉ G.edges then
                    createEdge(G, E_k)
                end if
                e ← getEdge(G, E_k)
                ojump ← n.octave - pn.octave
                id ← ojump + N_OCTAVES                   ▷ N_OCTAVES is 6 for a guitar
                e.ojumps[id] += 1
            end if
            pn ← n
        end for
    end for
end function
```

5.1 Networks Analysis

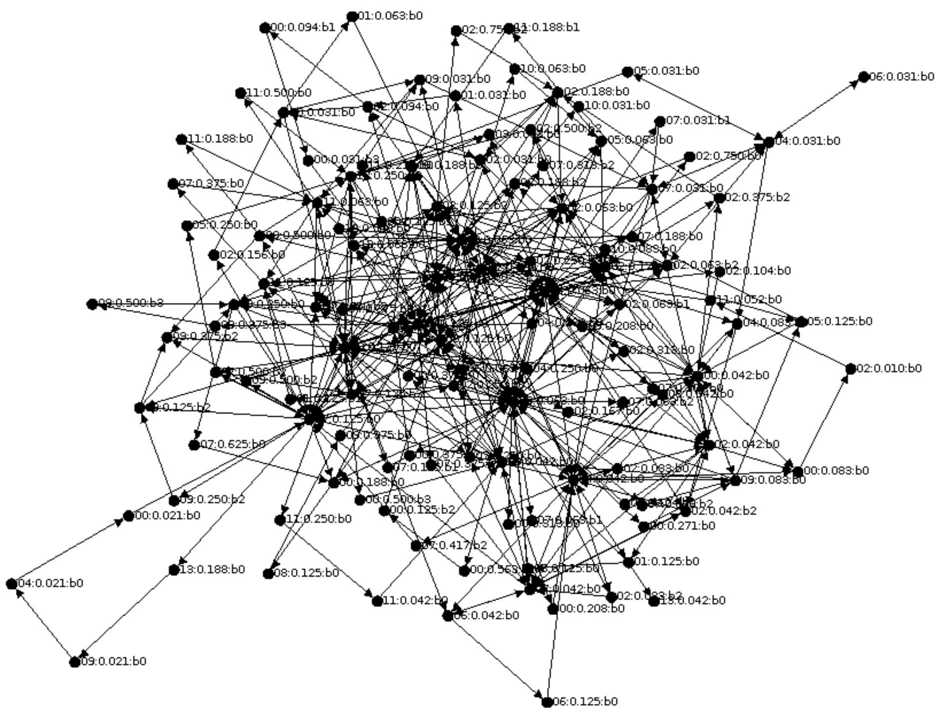

Fig. 2. A graph example created with 6 Jimi Hendrix solos. Edges of self linked nodes are not shown.

The figure 2 shows a graph created applying the algorithm 1 using 6 Jimi Hendrix solo files as input. The resulting graph counts 124 nodes and 484 edges and the bending effects have been considered. Interestingly, the same graph created without taking the bending effects into account counts 101 nodes. This seems to also happen with other graphs and the difference is greater for those guitarists that frequently resort to bendings. This suggests that considering more effects such as vibratos or virtual harmonics increases the vertex cardinality.

Many research articles about real network graphs, like biological, social and technological graphs, showed that they share a common feature: the so-called small-world (SW) property [8]. In SW networks most nodes can be reached from every other by a small numbers of steps and they are characterized by a high clustering coefficient. In order to verify if guitar licks networks respects the SW properties, the average clustering coefficient C and the average shortest path length L must be determined[9]. In this analysis the network is converted to a non-directed graph as a simplification. Let us start by defining the clustering coefficient: given e_{ij} an edge from the vertex v_i to the vertex v_j, the number of cycles of length 3 (closed triplets) from/to a node i is:

$$p_{ii}^{(3)} = \sum_{jk} e_{ij} e_{jk} e_{ki} ,$$

thus the local clustering coefficient of a node i is

$$C_i = \frac{p_{ii}^{(3)}}{k_i(k_i - 1)/2} ,$$

and the average clustering coefficient of a graph is

$$C = \frac{1}{N} \sum_i C_i$$

where N is the number of nodes in the graph. Moreover the shortest path length average L, is defined as:

$$L = \frac{1}{N(N-1)} \sum_{i,j,i \neq j} d_{ij}$$

where d_{ij} is the shortest path length from a node i to a node j. Then the table 1 shows the C and L values of some graphs created with the algorithm 1 and converted to non-directed graphs. In addition also the C_{rand} and L_{rand} values are represented. C_{rand} and L_{rand} values indicate the clustering coefficient and average path length of the equivalent derived random networks. The last column in the table refers to the small-coefficient σ defined as:

$$\sigma = \frac{C/C_{rand}}{L/L_{rand}}$$

What is remarkable is that the σ values are > 1, also $C \gg C_{rand}$ and $L \approx L_{rand}$, satisfying the SW property [9].

graph	N	C	C_{rand}	L	L_{rand}	σ
Dave Murray	102	0.329	0.074	2.864	2.602	4.039
Jimi Hendrix	124	0.429	0.064	2.675	2.649	6.63
Angus Young	122	0.266	0.068	3.092	2.888	3.65
Whole Graph	213	0.531	0.049	2.734	2.627	10.4

Table 1. Small World comparison results. Each graph is created with the tablature files of a particular guitarist except for the Whole Graph that is created with all the tablature files used for the other graphs generation.

6 Licks Classifier and Generator

As introduced before, a possible application for the graphs created with the method described in the previous section may be a lick classifier. Let us introduce the scenario in which a graph is created with the solos of different guitarists. It may be tempting to identify to which one of them an unknown guitar lick belongs to. This is possible because the links in the graphs are weighted according to the number of times they are walked through, meaning that if a sequence of musical notes has a higher total weight in a walk on a graph a than on a graph b, the probability that the sequence occurs more often on the graph a, and in the corresponding guitarist solos, than on the graph b is also higher. The lick classifier works like this: from a set of graphs and one lick it simply performs a walk of the lick on every graph collecting the corresponding total weights. The total weights are then dived by the number of notes used in the corresponding graphs creation.

Another proposed application is the Licks Generator: through the "LickNet" user interface, it is possible to select a graph from the graphs list and generate a set of licks which is sorted in a descending order by the matching lick score. Its algorithm is very simple: first it performs a very big number of random walks on the selected graph. Each of the walk has a resulting total weight with which the generator is able to sort them. Only the first N, N being a small number, licks are then considered to be significant for the selected graph.

It will be shown in the next section that even if the two applications described here are very simple, they can be effective.

7 Tests and Results

Table 2 shows the results of 4 tests made with the lick classifier. For convenience let us call *library* the set of graphs used in the experiments. Every graph of the library has been generated from a set of solos of a particular guitarist, namely Jimi Hendrix, Dave Murray and Angus Young.

For each experiment a guitar solo (licks set that was not used in the creation of any of the graphs) has been classified. Each row indicates a different experiment, the first column indicates the guitarists of the unknown licks set used for each experiment and the remaining columns indicate the scores gained for each graph by the corresponding unknown licks set.

Lick guitarist	Dave Murray	Jimi Hendrix	Angus Young
Jimi Hendrix	0.285	**0.356**	0.196
Dave Murray	**0.268**	0.052	0.040
B.B. King	0.011	**0.057**	**0.045**
Kirk Hammett	**0.724**	**0.785**	0.403

Table 2. Licks classifier test results.

Moreover the scores are relative to each test and should not be interpreted globally, as they are dependent on the length of a particular lick's notes sequence. It is interesting how well the classifier recognizes and classifies an unknown guitarist's licks set. What is also remarkable is that the matching expectation of the experiments are satisfied: without any surprise, the licks set of B.B. King is a closer match to the graph of Jimi Hendrix than the others. Also it is not very far from Angus Young while it is from the score of Dave Murray. This may be caused by the fact that Jimi Hendrix and Angus Young have been more influenced by the blues playing style. Furthermore it may seem weird that the licks set of Kirk Hammett, the lead guitarist of Metallica, is a closer match to Jimi Hendrix than the others. But it is also close to Dave Murray while it is farther away from Angus Young. This may be caused by the fact that Hammett was influenced by Jimi Hendrix in his early years [10] and that the genre played by Metallica is closer to the genre played by Iron Maiden than the genre played by AC/DC. In the end this is all speculation, a better interpretation would be desirable, also taking into consideration Kirk Hammett's graph, in order to determine the gap between those scores and the score achieved by Kirk Hammett himself. For example by the first two experiments one may interpret that Jimi Hendrix has a closer relation to Dave Murray than Angus Young, but with such limited amount of data this cannot be easily verified. A correlation metric among guitarists using my lick classifier may be modeled in future works.

The Licks Generator has been tested with the same library. For every guitarist a set of licks has been generated and tested again with the lick classifier in order to check wether the generated licks can be traced back to the respective guitarists. As expected, each set of generated licks gains the best score in correspondence to the right guitarist (Table 3).
In addition, in the LickNet source code repository[4] an example data folder has been provided, containing tablature files used to generate the library, tablature files of the licks used in the classifier tests and tablature files of the licks created with the lick generator.

Lick guitarist	Dave Murray	Jimi Hendrix	Angus Young
Jimi Hendrix	0.220	**<u>0.323</u>**	0.069
Dave Murray	**<u>0.293</u>**	0.208	0.052
Angus Young	0.117	0.131	**<u>0.449</u>**

Table 3. Results of the generated licks tested with the Licks Classifier.

[4] The repository is currently located at `https://github.com/matteomartelli/licknet`. Everything is free and open source. The instructions for contributing are provided in the README file.

8 Future Developments

The first one is to find a method to automatically retrieve and organise the tablature files. The main problem in this case is that these files are often badly written and most of them lack the key signature indication, therefore such a method can be hard to develop. Another solution may be to provide a public database of tablature files which have already been vetted, corrected and reorganised by their human users for usage by LickNet. As already introduced before, a second enhancement may be the implementation of a tool for correlating an artist with his licks and then comparing his style with other artists in the library simply evaluating the score gained for each corresponding graph.

Last but not least this software may be attached to other projects: for example it may be used in an improvisation software to generate notes for a soloist musician, or it may be attached to some tablature editor to easily generate licks as starting points for compositions.

9 Conclusions

Concerning complex network analysis, the networks of licks seem to satisfy the small world property. Furthermore, these networks seem to have a structure which is highly dependant from input data and from their corresponding guitarists. Therefore, functions that operate on the guitarists signature playing style, such as the licks classifier and the licks generator, stand to reason.

The initial expectations are well satisfied and LickNet appears to be a good start point for developing different kinds of software related to music classification and composition.

References

1. Maarteen van Steen, *Graph Theory and Complex Networks: An Introduction*, 2010.
2. R. Lambiotte, M. Ausloos, Uncovering collective listening habits and music genres in bipartite networks, *Phys. Rev. E 72, 066107*, 2005
3. Dr Tamás Nepusz, Reconstructing the structure of the world-wide music scene with Last.fm, http://sixdegrees.hu/last.fm
4. Benjamin Lind, Patterns in the Ivy: The Small World of Metal, http://badhessian.org/2013/09/patterns-in-the-ivy-the-small-world-of-metal/, 2013
5. Ramon Ferrer i Cancho, Ricard V. Solè, The small world of human language, *Proc. Royal Soc. London B, DOI: 10.1098/rspb.2001.1800* 2001
6. Daniel Mantilla, TuxGuitar: Editorial review, *Software Informer*, 2014.
7. Balakrishnan, V. K., *Graph Theory*, 1997.
8. Watts, D. J. and Strogatz, S. H., Collective dynamics of 'small-world' networks, *Nature 393, 440-442*, 1998
9. Qawi K. Telesford, Karen E. Joyce, Satoru Hayasaka, Jonathan H. Burdette, Paul J. Laurienti The Ubiquity of Small-World Networks, *Brain Connect.; 1(5): 367–375*, 2011

10. Brian Ives, Hendrix At 70: “My Favorite Guitar Player Of All Time” – Kirk Hammett Of Metallica, *CBS Local*, http://wcbsfm.cbslocal.com/2012/11/25/hendrix-at-70-my-favorite-guitar-player-of-all-time-kirk-hammett-of-metallica/, 2012