SIMILARITY OF STRUCTURES IN POPULAR MUSIC

BENOît CORSINI

Abstract. The study of the similarity matrix of a song has been a particularly efficient technique to characterize song structure. This method transforms a song into a matrix representing the proximity between its different sections. In this paper, these representations are used not to study the inherent structure of a song, but rather to compare them between each other. This allows the creation of metrics related to the pattern similarities of songs and opens the door to statistical studies on songs based on their pattern structure.

Contents

1. Introduction 1
2. The pattern similarity matrix 4
3. Comparing songs 5
4. Results 10
5. Discussion 16
Acknowledgements 19
References 19
Appendix A. Dataset statistics 20

1. Introduction

The study of the structure of music is a long lasting question that has been addressed in many ways. A common approach consists of grouping songs according to some specificity, such as genre and year, before analysing repetitions and common patterns in their composition [23]. This technique usually requires an in-depth analysis of the songs, using music theory, combined with human-based observations; examples of this technique can be found in [6] and [7], with the analysis of the structure of rock songs, in [8], where the structure of a specific song is being decomposed, or in [16], where a general structure is highlighted and corresponding songs are analyzed and given as examples. These methods however usually lack objectivity, and a more common approach consists of using algorithms based on mathematical models to analyze song structures [12]. The interest of such methods comes from their ability to highlight hidden properties that human-based approaches cannot find [4].

One approach to the study of music structure which shows promising results is to use 2-dimensional similarity matrices to represent songs [9]. This method is based on transforming a song into a matrix, whose entries correspond to the similarities between the different parts of the song. A common technique uses audio files and transforms them into sequences of frequencies; it then compares the frequencies between the different frames to compute the similarity matrix. The most common task performed using these matrices is the decomposition of songs into sections [1, 2, 9, 10, 18, 19]; however, they can also be used to extract information from the song [11], such as the tempo, or to distinguish the different signals in the song [20, 21], such as background and foreground voices. Similarity matrices have also recently been used to quickly identify different versions of the same song [22].

The study presented in this paper is based on the use of similarity matrices with two novel approaches. First, songs are analyzed not using a music file, but with a direct transcription of the music, using partitions of the different instruments. This allows a direct comparison of patterns in the song
and could open the door to more complex analysis, based on chord progressions for example. Second, the resulting similarity matrices are not used to analyse the structure of a specific song, but are compared between each other, leading to the definition of a distance on songs based on their patterns. This allows the characterization of groups of songs with similar structures and highlights common patterns which may arise by artist, years, or genres.

2-dimensional representation of songs. Consider a song whose structure falls into the typical verse-chorus-verse-chorus-solo-chorus progression. If comparing the different sections of the song, one could see that the verses and chorus repeat themselves. Moreover, if possible to quantify the difference between the sections, one could see that the solo is likely to be more different than the verses and the chorus. Representing these observations into a 2-dimensional matrix whose entries correspond to the general similarity between sections, something similar to Figure 1 would be obtained. Now, this representation is not strongly dependent on the song in terms of notes, chords, or tempo, but rather on its general structure, making it a useful tool to compare songs based on their pattern structure.

![Figure 1](image_url)  
**Figure 1.** A possible depiction of a song with the structure verse-chorus-verse-chorus-solo-chorus. White means that the two sections are the same, dark means that they are very different, and a scale of blue is used to interpolate in-between. In most songs, the solo is very different from the rest of the songs, whereas the difference between the verse and the chorus is not as significant. This representation only depends on the structure of the song and not on its notes, tempo, or genre.

The previous analysis and example uses a rather simple approach and divides a song into large sub-sections: verse, chorus, solo, etc. By doing a finer level of analysis, one could hope to highlight these macroscopic behaviours, as well as finer sub-patterns. Overall, the level at which the analysis is done can greatly influence the appearance of the results. In this study, since the representation is obtained from standard musical notations, a natural subdivision is the bar length. The types of results and figures obtained in this study can be summarized by Figure 2, which transforms the song *Beat It* by Michael Jackson into a 2-dimensional matrix where each entry describes the similarity between the bars of the song. Observe that the diagonal (top-left to bottom-right) is white since it corresponds to the similarity between the two same bars.

An important property for representing songs into matrices is that it is possible to transform abstract objects (songs) into mathematical objects (matrices), for which standard analysis techniques can be applied. This study uses this observation to define a distance $d_s$ on songs based on their pattern structure. This distance is meant to express the similarity of patterns between different songs. Using $d_s$, it becomes possible to identify groups of songs with similar structures.

Having defined $d_s$, this study will next extrapolate the statistical properties of song patterns. Combining $d_s$ with information on the songs, the following questions will be addressed.
**Figure 2.** The representation of the song **Michael Jackson - Beat It** and its structure decomposition. The song can be read in two directions: from left to right or from top to bottom, each square representing the similarity between two bars (see Section 2 for the precise definition of the matrix). Lighter squares correspond to smaller entries in the pattern matrix and darker squares correspond to larger entries, making the different sections of the song visible from the variation of shades.

- Given a song feature, such as year, genre or artist, do there exist specific feature values with a common pattern structure? Given a feature, what are the feature values with least variability in pattern structures?
- Given a group of songs with similar patterns, does there exist a corresponding feature value? Does this group correspond to a specific category of songs?

**Methodology.** This study aims to better understand the pattern structure of popular music, i.e. songs that at any point reached the *Billboard hot 100 chart* [3]. To compute the pattern matrices, all songs were used in the form of *Guitar Pro* files, which allowed for an analysis of the precise structure and composition of the songs. On top of the file, each song is defined by 6 features:

1. **artist:** the artist of the song.
2. **title:** the title of the song.
3. **year:** the first year the song appeared on the chart.
4. **decade:** the decade corresponding to the feature year.
5. **genre:** the genre(s) of the song.
6. **type:** types of genre, obtained from the feature genre.

The reason for considering the year and decade of first appearance on the chart rather than the actual release date is to focus on the evolution of patterns through popular taste, rather than when the songs were actually written and produced. The last feature, type, corresponds to general genres like rock, pop, or hip hop, as well as genre adjectives, such as experimental, classical, or psychedelic. The full list can be found in [5].
The dataset was collected using the *Billboard Hot weekly charts* dataset [14], which includes the first four features. A program [5] went through the songs of this dataset and collected the available Guitarpro files (exploring [24]). The *genre* feature was computed by searching for the song properties on *Wikipedia* [26] and the *type* feature was generated using the *genre* feature. More information on this dataset can be found in Appendix A and in [5].

Throughout this process, a lot of automation was used and this dataset may contain noise from any of the following steps:

- Errors in the original Billboard Hot dataset.
- Wrong song downloaded.
- Bad guitarpro file: fewer instruments, only partial song coverage, not original version, and mistakes in the writing process.
- Mismatched genres.

For each such step, human-based controls were put in place to reduce noise. Among these steps, the one which was the most difficult to control was the third step, related to the direct quality of the files. However, this type of noise should only concern a small portion of songs and will not strongly influence the analysis and the results.

2. The pattern similarity matrix

Let $S$ be a song divided into its ordered sequence of bars: $S = (B_k)_{1 \leq k \leq b}$. The *pattern similarity matrix* $P$ (or simply *pattern matrix*) is obtained by defining a distance $d_b$ on bars, used as the entries of $P$:

$$P_{k,\ell} = d_b(B_k, B_{\ell}).$$

If the song is composed of different instruments, the pattern matrix will be computed by summing the contributions of the different instruments:

$$P_{k,\ell} = \sum_{i \in \text{Ins}(S)} d_b(B^{(i)}_k, B^{(i)}_{\ell}),$$

where Ins($S$) is the set of instruments of $S$ and $B^{(i)}_k$ is the $k$-th bar of instrument $i$.

**Distance of similarity on bars.** In order to define $d_b$, bars need to be transformed. The choice of transformation is to consider them as vectors of size $s$, where $s$ is the number of time subdivisions in the bar. In other words, a bar $B$ is defined as a sequence $B = (N_t)_{1 \leq t \leq s}$ where each element (or note) $N_t$ corresponds to what is being played at time $t$, and $s$ corresponds to the product of the smallest note length and the ratio of the bar time signature. An example of such representation can be found in Figure 3, using the main riff of *Seven Nation Army* by *White Stripes*.

Let $B = (N_t)_{1 \leq t \leq s}$ and $\hat{B} = (N_t)_{1 \leq t \leq s}$ be two bars. A natural definition for $d_b$ is to count the number of differences between $B$ and $\hat{B}$:

$$d_b(B, \hat{B}) = \sum_{1 \leq t \leq s} \delta(N_t, \hat{N}_t),$$

where

$$\delta(N, \hat{N}) = \begin{cases} 
0 & \text{if } N \text{ and } \hat{N} \text{ are the same} \\
1 & \text{if exactly one of } N \text{ and } \hat{N} \text{ is empty} \\
2 & \text{otherwise}
\end{cases}.$$

With this choice of $d_b$ and $\delta$, each time a note is being played in one bar, it adds one to their distance, except if the same note is being played at the same time. From this definition, $d_b(B, \hat{B}) \in \{0, 1, \ldots, 2s\}$, which implies that the range of $d_b$ is related to the number of notes per bar.

In the definition of $d_b$, it is assumed that the two bars have the same size $s$. This does not necessarily follow from the previous definition of $s$, but this can be obtained by normalizing the two bars using the smallest note subdivision and the longest time signature ratio of both bars. With this definition, if the two bars have different time signatures, then the matrix representation of the bar with smaller
Figure 3. The representation of the four bars \((B_1, B_2, B_3, B_4)\) into four vectors of size \(s = 16\). This choice for \(s\) follows from the smallest subdivision being a sixteenth note (due to the dotted eighth note) and the bars having a time signature of 4/4, meaning that there is at most \(16 \times 4/4 = 16\) notes in each bar. For each time \(t \in \{1, \ldots, 16\}\), the entry \(t\) of the vector corresponds to the note being played at time \(t\), and is empty if no note is being played. A special letter, \(R\), is used for a rest.

Figure 4. An example of two bars with different time signatures. If computing \(d_b\) for this pair of bars, the result would be 1 since the first four notes are the same and the last note of the second bar corresponds to a time which does not exist in the first bar. This choice comes from the fact that longer bars are usually used to extend a previous pattern (and similarly, shorter bars are usually used to reduce a previous pattern). This choice, however, does not have a significant effect on the results since most of the songs considered here have a classical 4/4 time signatures.

With \(d_b\) being defined for any pair of bars, the pattern matrix \(P\) of any song can now be computed. As an example, the pattern matrix of the main riff of Seven Nation Army, whose vector representation was given in Figure 3, can be found in Figure 5. For the rest of the paper, the pattern matrix of a song refers to the matrix obtained using this definition of \(d_b\).

3. Comparing songs

With the pattern matrix of a song defined in Section 2, a distance on pattern matrices can now be computed in order to obtain \(d_s\), the distance on song structures. This section focuses on the definition of \(d_s\).

Directly comparing pattern matrices presents a few problems. First of all, they do not necessarily all have the same shape, as \(P\) has size \(b \times b\), where \(b\) is the number of bars of the song. Second, recall that the entries in these matrices are related to the number of notes being played, implying that they can greatly vary between songs. Finally, this link between entries of the pattern matrix and number of notes played can also create high variability of entries inside a single song, for example if a sub-section has a much higher density of notes (such as a solo, see Figure 1).

For the rest of this section, the goal is to define the function

\[
\nu : P \in \mathbb{N}^{b \times b} \longrightarrow \nu(P) \in [0, 1]^{s_0 \times s_0}
\]
<table>
<thead>
<tr>
<th>pair of bars</th>
<th>distance</th>
<th>vector representation of the two bars</th>
<th>depiction of the distance in the original music sheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 2)</td>
<td>8</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
<tr>
<td>(1, 3)</td>
<td>0</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
<tr>
<td>(1, 4)</td>
<td>10</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
<tr>
<td>(2, 3)</td>
<td>8</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
<tr>
<td>(2, 4)</td>
<td>2</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
<tr>
<td>(3, 4)</td>
<td>10</td>
<td>v1 v2 v3 v4 v5 v6</td>
<td><img src="image" alt="depiction of distance" /></td>
</tr>
</tbody>
</table>

**Figure 5.** A representation of the entries of the pattern matrix of the main riff of *Seven Nation Army* by *White Stripes*. For each row, the first column gives the indices of the two bars being compared, the second column gives the distance between these two bars, the third column gives the two vector representations of the bars obtained in Figure 3, and the last column gives the original music sheet used to compute the pattern matrix. The notes highlighted in yellow in the third and fourth columns correspond to the differences between the two bars and each yellow square counts as one in the distance of the two bars. From this figure, one can see that the corresponding pattern matrix is given by

$$
\begin{bmatrix}
0 & 8 & 0 & 10 \\
8 & 0 & 8 & 2 \\
0 & 8 & 0 & 10 \\
10 & 2 & 10 & 0
\end{bmatrix}
$$

used to normalize pattern matrices so that they can be easily compared with each other. Here, $s_0$ is a constant over all songs and $\nu(P)$ is a normalized pattern matrix and should reflect the pattern structure of $P$ while avoiding the previously raised issues. With $\nu$ being defined, the distance over songs $d_s$ is set as

$$
d_s(S_1, S_2) = \|\nu(P_1) - \nu(P_2)\|_p,
$$

where $P_1$ and $P_2$ are the pattern matrices of $S_1$ and $S_2$. Here, $\| \cdot \|_p$ is defined as the average $L^p$ norm; that is, for all $A$ of size $s_0 \times s_0$, we have $\|A\|_p = \left(\frac{1}{s_0^2} \sum_{1 \leq k, \ell \leq s_0} |A_{k,\ell}|^p\right)^{1/p}$. The value of $p$ is implied in the definition of $d_s$ and set to $p = 2$ for the rest of this study.

**Pattern matrix normalization.** Let $P$ be a pattern matrix of size $b \times b$ whose entries take values in $\mathbb{N}$. In order to visualize the impact of the different modifications made on $P$ by the normalizing function $\nu$, matrices are depicted as in Figure 2: a white square corresponds to a 0 in the matrix, whereas a black square corresponds to the maximal possible entry, and a scale of blue is used to interpolate values in-between.

As explained earlier, the entries of $P$ are related to the number of notes being played, which means that these entries can greatly vary between songs. In order to avoid scaling problems when comparing two pattern matrices, their entries have to be taking values in $[0, 1]$. The importance of this constraint is represented in Figure 6, where the two songs represented have similar structures but different ranges of entries in their pattern matrices.

The problem of large entries in pattern matrices does not simply affect comparing two different songs, but can also prevent the matrix of a given song from clearly highlighting its structure. For example, a song with a section denser than the other ones will have very high entries in specific
Figure 6. Comparison of the songs Jonas Brothers - Lovebug and Metallica - One before and after normalizing their entries between 0 and 1. Each column represents the two pattern matrices on the same scale of blue; the left column corresponds to the original pattern matrices, whereas the right column corresponds to the pattern matrices whose entries have been scaled between 0 and 1. As can be observed from the right column, these two songs have a similar structure. However, if only comparing the values of the two pattern matrices (left column), the entries of the matrix of the Metallica song are much larger than the entries of the matrix of the Jonas Brothers song.

rows and columns of its pattern matrix, making the rest of the entries small in comparison; and this could hide possible sub-patterns of the song. To avoid this issue, the variation of the entries in $P$ are normalized by using the corresponding percentile index rather than the exact value $1$.

Write $\tilde{P} = (P_{k_i,\ell_i})_{1\leq i \leq b^2}$ for the sequence of ordered entries of $P$; in other words, $(k_i, \ell_i) \neq (k_j, \ell_j)$ for $i \neq j$, and $P_{k_i,\ell_i} \leq P_{k_{i+1},\ell_{i+1}}$. Let $q_\alpha = P_{k_{\alpha b^2},\ell_{\alpha b^2}}$ be the $\alpha$ quantile of $\tilde{P}$ and $p_j = q_{j/100}$ be the $j$-th percentile of $\tilde{P}$. The percentile normalization function $\nu_p$ is defined as

$$\nu_p : P \mapsto \left( \max\{j : p_j \leq P_{k,\ell}\}\right)_{1 \leq k,\ell \leq b},$$

and replaces the entries of $P$ by their corresponding percentile index. An example of the importance of applying $\nu_p$ can be found in Figure 7.

After applying $\nu_p$ to a pattern matrix $P$, the maximal entry is always 100, corresponding to the maximal entry or entries of $P$. However, the minimal entry might not be 0, for example if there are many 0 in $P$. In order to avoid this issue, a second normalizing function $\nu_0$ is applied, forcing the entries of the matrix to take values in $[0, 1]$, with 0 and 1 included. More precisely $\nu_0$ is defined as

$$\nu_0 : \tilde{P} \mapsto \frac{1}{100 - \min\{\tilde{P}\}} (\tilde{P} - \min\{\tilde{P}\}),$$

where $\{\tilde{P}\} = \{\tilde{P}_{k,\ell}, 1 \leq k,\ell \leq b\}$. Here, $\tilde{P}$ is used instead of $P$ to highlight the fact that $\nu_0$ is supposed to be applied to the image of $P$ through $\nu_p$ and not directly to the pattern matrix $P$. The effect of

$1$The reason for choosing the percentile index instead of a more common activation function, such as softmax or softmin, comes from the desire to completely remove the bias created by large entries in the matrix, instead of just reducing it.
Figure 7. Comparison of the songs Guns N’ Roses - Sweet Child O’ Mine and Dire Straits - Sultans of Swing with and without applying $\nu_p$ to their pattern matrices. The left column corresponds to the original pattern matrix and the right column corresponds to its image through $\nu_p$. As can be observed from the left column, a few bars with high distance to the rest of the song tend to make the overall picture uniformly blue, making both of these pattern matrices look similar. However, after applying $\nu_p$, different patterns appear more clearly, showing the diversity in structure of these two songs. This figure also represents the importance of $\nu_p$ when representing patterns of songs, since it highlights their patterns.

$\nu_0$ on $\tilde{P}$ is depicted in Figure 8. The changes operated by $\nu_0$ are not meant to have a strong impact on the whole process, but give a desired property for the resulting matrix: bars which are the same correspond to entries with value 0, and bars with most differences correspond to entries with value 1.

Figure 8. Comparison of $\nu_p(P)$ and $\nu_0(\nu_p(P))$ for the pattern matrix $P$ of the song Michael Jackson - Smooth Criminal. The main difference between these two pictures is that the lightest colour on the left image is not exactly white, meaning that the smallest entries of the matrix are not exactly 0. Even though $\nu_0$ does not greatly modify the matrix, the patterns appear slightly clearer on the right.

When applying $\nu_p$ and then $\nu_0$ to a pattern matrix $P$, it is transformed into a matrix of size $b \times b$ whose entries take values in $[0, 1]$. Moreover, the resulting matrix clearly represents the structure of
Indeed, if \( \tilde{d} \) corresponds to the Riemann approximation of \( d \), as are related to the entries of \( \tilde{d} \) by a function similar to \( \sigma \). Let \( \nu_0 \cdot \nu_p \). A natural way to compare \( \tilde{P} \) and \( \tilde{Q} \) in spite of their respective sizes is to extend them by repeating each of their entry multiple times. More precisely, \( \tilde{P} \) and \( \tilde{Q} \) are transformed using \( \sigma \), defined as follows:

\[
\sigma : (\tilde{P}, \tilde{Q}) \mapsto \left( (\tilde{P}_{k/d}, \lceil t/d \rceil)_{1 \leq k, t \leq bd}, (\tilde{Q}_{k/b}, \lceil t/b \rceil)_{1 \leq k, t \leq bd} \right).
\]

The image of \((\tilde{P}, \tilde{Q})\) through \( \sigma \) is a pair of matrices, whose sizes are both \( bd \times bd \), and whose entries are related to the entries of \( P \) and \( Q \) repeated multiple times: \( d^f \) times for the entries of \( P \) and \( b^2 \) for the entries of \( Q \). Using \( \sigma \), let the true distance \( d^t \) of \( P \) and \( Q \) be defined by:

\[
d^t(\tilde{P}, \tilde{Q}) = \left\| \tilde{P}^\sigma - \tilde{Q}^\sigma \right\|_p,
\]

where \((\tilde{P}^\sigma, \tilde{Q}^\sigma) = (\sigma(\tilde{P}), \sigma(\tilde{Q})) \). Note that this distance also corresponds to a graphon-like approach \([13]\). Indeed, if \( \tilde{P} \) and \( \tilde{Q} \) are transformed into functions \( f^{\tilde{P}}, f^{\tilde{Q}} : [0, 1]^2 \mapsto [0, 1] \) such that \( f^{\tilde{P}}(x, y) = \tilde{P}_{[xb], [yk]} \) and \( f^{\tilde{Q}}(x, y) = \tilde{Q}_{[xd], [yd]} \), then \( d^t \) is equal to their integral \( L_p \) distance

\[
d^t(\tilde{P}, \tilde{Q}) = \left( \int_{(x,y)\in[0,1]^2} \left| f^{\tilde{P}}(x, y) - f^{\tilde{Q}}(x, y) \right|^p dxdy \right)^{\frac{1}{p}}.
\]

This approach reinforces the idea that this should be the true distance between pattern matrices.

The function \( d^t \), although perfectly adapted to compute the distance between normalized pattern matrices, is not computationally feasible since it requires to use matrices of size \( bd \times bd \); even if replacing \( bd \) by the lowest common multiple of \( b \) and \( d \), the computation time is unrealistic (see Table 1). To avoid this computational problem, an approximation of \( d^t \) is used.

Fix an integer \( s_0 \geq 1 \) and let \( \nu_{s_0} \) be defined as follows:

\[
\nu_{s_0} : \tilde{P} \mapsto (\tilde{P}_{k/s_0}, \lceil t/s_0 \rceil)_{1 \leq k, t \leq s_0}.
\]

This function, similar to \( \sigma \), creates a matrix of size \( s_0 \times s_0 \) whose entries are related to the entries of \( \tilde{P} \). However, since \( s_0 \) might not be a multiple of \( b \), the number of times each entry of \( \tilde{P} \) is repeated in \( \nu_{s_0}(\tilde{P}) \) is not necessarily constant, creating a small error in the representation of \( \tilde{P} \). Define now \( d^{s_0} \) by:

\[
d^{s_0}(\tilde{P}, \tilde{Q}) = \left\| \nu_{s_0}(\tilde{P}) - \nu_{s_0}(\tilde{Q}) \right\|_p.
\]

Since \( \nu_{s_0}(\tilde{P}) \) slightly modifies the role of the entries of the matrix \( \tilde{P} \), the distance \( d^{s_0} \) is not necessarily going to be equal to \( d^t \). However, for any pair of matrices \( \tilde{P} \) and \( \tilde{Q} \), the distance \( d^{s_0}(\tilde{P}, \tilde{Q}) \) converges to \( d^t(\tilde{P}, \tilde{Q}) \) as \( s_0 \) increases to infinity; if considering the integral definition of \( d^t \) as in (2), \( d^{s_0} \) actually corresponds to the Riemann approximation of \( d^t \) using squares of size \( \frac{1}{s_0} \times \frac{1}{s_0} \).

Computing \( d^{s_0} \) requires using matrices of size \( s_0 \times s_0 \). This means that the smaller \( s_0 \) is, the faster the computation will be. Hence, it is interesting to choose a value for \( s_0 \) which balances between computational time of \( d^{s_0} \) and accuracy compared to the true distance \( d^t \). Table 1 gives an approximation of the computational time and the accuracy for different values of \( s_0 \). From this table, any \( s_0 \) larger than 100 is an appropriate choice since the error rate is less than 1%. For the rest of this study, \( s_0 \) is set to be 500, corresponding to an error rate as small as computation time reasonably allows.

To conclude this section, \( \nu \) is naturally set to be a combination of the different normalization functions:

\[
\nu : P \mapsto \nu_{s_0}(\nu_0(\nu_p(P))).
\]

In other words, the image of a pattern matrix \( P \) through \( \nu \) corresponds to transforming \( P \) using its percentiles (see Figure 7), then forcing its values to span 0 and 1 (see Figure 8), and finally resizing it.
<table>
<thead>
<tr>
<th>distance</th>
<th>(d^l)</th>
<th>(d^{100})</th>
<th>(d^{200})</th>
<th>(d^{1000})</th>
<th>(d^{2000})</th>
</tr>
</thead>
<tbody>
<tr>
<td>error</td>
<td>0%</td>
<td>0.99%</td>
<td>0.50%</td>
<td>0.20%</td>
<td>0.10%</td>
</tr>
<tr>
<td>time (ms)</td>
<td>6318</td>
<td>2.43</td>
<td>13.5</td>
<td>43.0</td>
<td>164</td>
</tr>
</tbody>
</table>

Table 1. Comparison of \(d^l\) with \(d^{s_0}\) for \(s_0 \in \{100, 200, 500, 1000, 2000\}\). For a given distance \(d^{s_0}\), its error corresponds to the average error rate between \(d^l(\tilde{P}, \tilde{Q})\) and \(d^{s_0}(\tilde{P}, \tilde{Q})\) (where the error rate between \(a\) and \(b\) is defined as \(|a - b|/|a|\)). For each distance \(d^l\), its time corresponds to the average time to compute \(d(\tilde{P}, \tilde{Q})\). All these results were approximated by randomly choosing 400 pairs of songs from the dataset and averaging their error rate and time to compute.

to \(500 \times 500\). Using the image of \(\nu\) as a normalized pattern matrix, the pattern structure of songs can be compared by defining the similarity distance on songs \(d_s\) as in (1).

4. Results

With the pattern matrix being defined in Section 2 and the distance \(d_s\) set in Section 3, songs can now be compared according to their pattern structures. In this section, the values given by \(d_s\) are combined with the features of the songs (defined in Section 1) to answer the previously asked questions:

- Given specific feature values, what can we say about the pattern structures?
- Given similar pattern structures, do they have corresponding feature values?

To answer these questions, two scores are defined on groups of songs: the pattern variability score (\(V^F\) score) and the feature variability score (\(V^P\) score).

Let \(G_s = \{S_1, \ldots, S_k\}\) be a group of songs. The \(V^P\) score of \(G_s\) is defined as the average distance between any pair of songs of \(G_s\):

\[
V^P = \mathbb{E}_{S \neq \tilde{S} \in G_s} \left[ d_s(S, \tilde{S}) \right] = \frac{1}{k(k-1)} \sum_{1 \leq i, j \leq k} d_s(S_i, S_j).
\]

This score should be interpreted as the amount of variability between the patterns of the songs of this group: groups with a small \(V^P\) score correspond to a set of songs whose patterns are similar.

Given this same group of songs \(G_s = \{S_1, \ldots, S_k\}\), let \(f_1, \ldots, f_k\) be their feature values, corresponding to the feature \(F\) taken from the possible choices of features: artist, title, year, decade, genre, or type. The \(V^F\) score of \(G_s\) (with regards to \(F\)) is defined in one of the following two ways. If \(f_1, \ldots, f_k\) are real numbers (meaning that \(F \in \{\text{year, decade}\}\)), then the \(V^F\) score is the standard deviation of the values \(f_1, \ldots, f_k\):

\[
V^F = \sqrt{\frac{1}{k} \sum_{1 \leq i \leq k} (f_i - \mu)^2},
\]

where \(\mu = \frac{1}{k} \sum_{1 \leq i \leq k} f_i\). Otherwise, if \(f_1, \ldots, f_k\) are not real numbers, then the \(V^F\) score is defined as the average of the counting process of \(f_1, \ldots, f_k\). More precisely, let \((c_1, \tilde{f}_1), \ldots, (c_\ell, \tilde{f}_\ell)\) be such that \(c_j = \#\{i : f_i = \tilde{f}_j\}\), \(c_1 \geq \ldots \geq c_\ell\) and \(\{f_1, \ldots, f_k\} = \{\tilde{f}_1, \ldots, \tilde{f}_\ell\}\). Then, the \(V^F\) score is defined as

\[
V^F = \frac{1}{c_1 + \ldots + c_\ell} \sum_{1 \leq j \leq \ell} j c_j.
\]

This choice of score can be seen as transforming \(f_1, \ldots, f_n\) into a set of probabilities \(\{p_1, \ldots, p_\ell\}\), where \(\ell\) is the number of different feature values in \(\{f_1, \ldots, f_n\}\) and \(p_j\) is the probability of having the \(j\)-th most common feature value when sampling a random \(f_i\). In both cases, the \(V^F\) score should be interpreted as the amount of variability between the feature values of the songs of this group: groups with a small \(V^F\) score correspond to a set of songs with a lot of similar features.
With the $V^P$ and the $V^F$ scores defined, the rest of this section is organized as follows. Section 4.1 uses a feature-based approach and studies the $V^P$ score when grouping songs according to their feature values. For each feature, this allows a ranking of the feature values according to their variability of patterns. Section 4.2 and 4.3 create groups of songs according to the distance $d_s$ and study the $V^F$ score of these groups. Section 4.2 uses a cluster-based approach and partitions songs into clusters, whereas Section 4.3 uses a neighbour-based approach and focuses on each song’s nearest neighbours. These two approaches give similar results and are both efficient in finding underlying properties of features not observable with the approach of Section 4.1.

4.1. Pattern variability on features (feature-based approach). Consider the different features of a song: artist, title, year, decade, genre and type. Each of these features has a set of possible values it can take; for example, all songs of the dataset in this study appeared in the chart for the first time between 1958 and 2019. Now, for each feature value, all corresponding songs can be grouped together and their $V^P$ score can be computed. For example, the first 20 lowest $V^P$ scores for the feature year are represented in Figure 9.

![Figure 9. The year 1958 to 2019 ordered according to their $V^P$ score (only the first 20 lowest ones are depicted here). The blue dots corresponds to the $V^P$ scores, which are obtained by computing the average distance between songs from a given year. The blue bars show the standard deviation around this average distance. The bars in red represent the number of songs who appeared for the first time the corresponding year. No year has a notably lower score than the next one, but this computation gives a ranking on years according to their variability in pattern structure.](image)

The $V^P$ score of each feature value is a clear and easy to interpret metric. However, this score is difficult to combine with the actual pattern matrices, since it is reflecting a general trend rather than the existence of a unique pattern structure. Indeed, when the number of songs for each feature value is large, then the number of pattern structures tends to be large as well. Figure 10 illustrates this principle by representing the artist with at least 2, 5 and 10 songs and with minimal $V^P$ score. From this figure, one can see that, as the number of songs considered increases, the number of pattern structures also naturally increases.

A possible explanation for the limited interpretation of $V^P$, is that it only focuses on the average distance between a given group of songs $G_s$. As explained, this means that, as the size of the group
Figure 10. The artists with lowest $V^P$ score according to different threshold on their minimal number of songs. On the left, three figures correspond to the first 20 scores for the feature artist, where the minimal number of songs is set to 2, 5 and 10. On the right, all the normalized pattern matrices of each artist with lowest score. As the number of songs for each feature value increases, so does the variety of patterns, making it more difficult to explain the low score with the pattern matrices.
require some modification and improvement (see Section 5 for further ideas), but could be used as an interesting metric to study the evolution of variability through years, genres, or artists.

4.2. Feature variability on clusters (cluster-based approach). After considering each feature independently and giving a score for each feature value according to pattern variability, this section applies first the distance on song $d_s$ to create clusters. Once these clusters are identified, their $V^F$ score is computed, highlighting clusters with repeating feature values.

Since clusters can greatly vary according to the method used, three common clustering algorithms were implemented:

- Spectral Clustering [15];
- K-Medians [17]; and
- Agglomerative Clustering [25].

The choice of algorithms was motivated by two constraints. First, these algorithms can be directly applied to a pre-computed distance matrix obtained from $d_s$, and do not need to access the embedding space of the songs; this first reason explains why K-Medians was chosen over the more classical K-Means. Second, the number of output clusters is determined by a number given as input; this second reason explains why the classical DBSCAN algorithm is not used here. With these two constraints, Spectral Clustering and K-Medians are standard algorithms. The reason for using Agglomerative Clustering is that it fits the songs into a tree-like structure, which is a good way to interpret song patterns (see Figure 11). For this last reason, Agglomerative Clustering gives the best results and all further analysis was obtained using this algorithm.

Since the dataset contains a large number of songs (4166 in total, see Appendix A), and the size of the clusters should be limited for interpretative reasons, a recursive approach was used. The actual clustering algorithm is described in Algorithm 1 and is based on applying one of Agglomerative Clustering, Spectral Clustering, or K-Medians recursively, to reduce the clusters to a desirable size. The results that follow are all obtained from using Agglomerative Clustering (with $n_c = 2$ output clusters), no limit on the maximal number of iterations ($m_i = |G_s|$), and clusters of size at most 15 ($m_s = 15$).

Once Algorithm 1 is applied to the set of songs, a similar approach to Section 4.1 is used and clusters are ranked according to their $V^F$ score. A typical outcome of this algorithm can be found in Figure 12, where clusters are ordered according to their $V^F$ score with regards to the feature year. Once this figure is obtained, it needs to be combined with an analysis of the clusters; the cluster with lowest score in Figure 12 is represented in Figure 13.

The interest of using clusters instead of the feature-based approach of Section 4.1 is natural when considering results of Figures 12. Indeed, the feature-based approach considers every feature value independently of each other and then ranks them according to their $V^P$ score. However, in the case of the feature year for example, feature values can be compared between each other and a notion of proximity can be defined. Using this property, the cluster represented in Figure 13 can be found, where the pattern clearly corresponds to songs around the year 2010. This could not have been unveiled by the feature-based approach in Section 4.1 since it does not compare songs that have different feature values.

The reason for using the cluster-based approach over the feature-based approach is not limited to the notion of proximity on the feature year. As explained in Section 4.1, a possible limitation of the $V^P$ score is that it is not able to detect groups of songs that would be generated from a couple of distinct pattern structures. Clustering methods, however, are able to separate songs according to pattern similarity and then identify groups with repeated feature values. An example of the power of clusters in identifying sub-groups of patterns inside a given feature value is represented in Figure 14, where the artist Linkin Park shows two commonly used structures. The reason for this artist not to appear clearly in the results of Figure 10 comes from the two limitations of the $V^P$ score explained in Section 4.1: this artist has a lot of songs and a combination of different but repeated pattern structures.

Using the $V^F$ score on clusters shows more promising results than using the $V^P$ score on songs grouped by their feature values, since it is able to highlight finer properties of song patterns. One reason for this improvement is that the cluster-based approach is able to ignore noisy or out of distribution
Figure 11. A representation of the tree-like structure of song patterns. This graph was obtained by connecting songs with their nearest neighbours. From this figure, one can observe that all of the above songs fall into the category of songs with a high *outro*, represented by darker bands at the bottom and at the right of the figures. From this general pattern structure category, sub-types of patterns can be observed and the connections between these patterns create a tree-like structure.

songs and only focuses on groups with similar structures. This leads to a better ability to identify specific pattern structures related to years (Figure 13) or artists (Figure 14).

4.3. Feature variability on neighbourhoods (neighbour-based approach). The cluster-based approach used in Section 4.2 is useful since it creates a partition of the songs into groups. This approach also implies that groups with low $V_F$ score can be interpreted as patterns characteristic of the given feature value (as explained in Figure 13). However, another possible approach on grouping songs is to consider the set of nearest neighbours for each song.

For each song in the dataset, using the distance $d_s$ it is possible to identify its neighbourhood, hence creating groups of songs with a centre. With this approach, some songs might appear in more neighbourhoods than others. This remark implies that neighbourhoods cannot be interpreted as characteristic groups, like it was the case with clusters; however, this technique allows the definition of a centre song, opening more possibilities with the analysis of features.
Algorithm 1: Recursive clustering algorithm

\begin{algorithm}
\caption{Recursive clustering algorithm}
\begin{algorithmic}
\STATE \textbf{input} : A group of songs $\mathcal{G}_s$ and a metric $d_s$
\STATE \textbf{output} : A cluster partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of $\mathcal{G}_s$
\STATE \textbf{params:}
\STATE \quad • $A_C$, a clustering algorithm, with a given number $n_c$ of output clusters
\STATE \quad • $m_s$, the maximal size of a cluster
\STATE \quad • $m_i$, the maximal number of iterations
\begin{algorithmic}
\STATE \textbf{begin}
\STATE \quad $\mathcal{C} \leftarrow \{\mathcal{G}_s\}$
\STATE \quad \textbf{for} $i \leftarrow 1$ \textbf{to} $m_i$ \textbf{do}
\STATE \quad \quad $\mathcal{C}_{\text{aux}} \leftarrow \emptyset$
\STATE \quad \quad \textbf{for} $C \in \mathcal{C}$ \textbf{do}
\STATE \quad \quad \quad \textbf{if} $|C| > m_s$ \textbf{then}
\STATE \quad \quad \quad \quad $\mathcal{C}_{\text{aux}} \leftarrow \mathcal{C}_{\text{aux}} \cup A_C(C, d_s)$\quad \textit{/* $A_C(C, d_s)$ partitions $C$ in $n_c$ clusters using $d_s$ */}
\STATE \quad \quad \quad \textbf{else}
\STATE \quad \quad \quad \quad $\mathcal{C}_{\text{aux}} \leftarrow \mathcal{C}_{\text{aux}} \cup \{C\}$
\STATE \quad \quad \textbf{end}
\STATE \quad \textbf{end}
\STATE \quad $\mathcal{C} \leftarrow \mathcal{C}_{\text{aux}}$
\STATE \textbf{end}
\STATE \textbf{return} $\mathcal{C}$
\end{algorithmic}
\STATE \textbf{end}
\end{algorithmic}
\end{algorithm}

Consider computing the 20 nearest neighbours of all the songs. For each of these neighbourhoods, the corresponding $V^F$ score can be computed as in Section 4.2. Furthermore, since neighbourhoods have a centre song, it is now possible to compare the feature of the centre with its neighbours. By applying this idea to the neighbourhoods ordered according to their $V^F$ score with regards to the feature year, Figure 15 is obtained and compares the neighbours and the centre song years of first appearance in the chart.

An interesting analysis on the feature year, made possible by neighbourhoods, is to classify song patterns as one of early, late or on-time (see Figure 15). By considering neighbourhoods with lowest $V^F$ with regards to the feature year, meaning neighbourhoods with lowest year standard deviation, and comparing the average year of the neighbourhood to the year of the centre song, three behaviours can be identified. Songs can either have pattern structures similar to songs appearing later in the chart, meaning that this pattern was ahead of its time. Similarly, songs can have pattern structures that were commonly used in earlier years, meaning that it could have been inspired by previously released songs. Finally, songs can have pattern structures similar to other songs of the same year. In this last case, it is possible that the cluster-based approach of Section 4.2 would have already identified this relation between pattern and years, whereas the first two cases would tend to be hidden when studying the $V^F$ score of the clusters. An interesting example of a song with late pattern structure can be found in Figure 16, where on top of a difference in years, there is also a difference in genre.

Overall, the cluster-based approach of Section 4.2 and the neighbour-based approach of this section give similar and complementary results. While the cluster-based approach highlights characteristic patterns corresponding to feature values, the neighbourhood approach can be used to find outliers in pattern structures by showing songs whose features are unrelated to those of their neighbours. These two techniques can also be combined to help understand the general structure of the song patterns. Indeed, by comparing clusters and neighbourhoods, one can observe a possible tree-like structure for song patterns, as highlighted in Figure 11. This might also explain why the Agglomerative Clustering algorithm shows better results than the two other clustering techniques considered.
Figure 12. Clusters ordered according to their $V_F$ score with regards to the feature year (only depicting the 20 lowest scores). The blue dots represent the average year of the cluster, the blue bars represent the standard deviation around this year and the red bars represent the sizes of the clusters. This figure needs to be combined with a representation of the clusters in order to appreciate its results. The cluster with the lowest score is represented in Figure 13.

Figure 13. A representation of the cluster with lowest $V_F$ score as found in Figure 12. As one can see, all songs show a similar pattern and have been released around the same years. Since this cluster was obtained by grouping songs according to their pattern similarities, it also means that this specific pattern is characteristic of the end of the 2000's.

5. Discussion

Embedding songs into 2-dimensional similarity matrices is a powerful tool in studying their underlying structure. Moreover, this method can be applied to compare songs between each other according to their pattern structures. This idea allows the definition of precise metrics on groups of songs which then help to unveil underlying properties of artists, years, or genres. By defining the similarity matrices and the distance on these matrices, one can observe interesting repetitions and commonly used structures over a large set of songs.

The most natural approach is to group songs according to some known information, such as by artist, or year, before comparing their pattern structures. This leads to the definition of the $V_F$ score and the study of Section 4.1. Although theoretically interesting, this method shows limitations due to the inevitable variety of structures which appear when comparing multiple songs (see the analysis made in Figure 10). To avoid such problems, two possible improvements on the $V_F$ score could be implemented.
FIGURE 14. A representation of all the pattern matrices of the artist Linkin Park. The top two groups each correspond to a specific type of pattern and the bottom group contains the rest of the songs. This grouping shows that the artist Linkin Park commonly uses one of the two top structures. This property is highlighted by clusters but was missed when considering the $V_P$ score on the artist as shown in Figure 10. Two reasons can explain why this artist does not have a low $V_P$ score: the large number of songs it has in the dataset, and the fact that they mostly correspond to one of two distinct pattern structures.

The first improvement would be to normalize all groups to the same size. This could be done either by changing the dataset directly, or by computing differently the $V_F$ score: for example, instead of summing the average distance over all songs, just consider the $N$ closest songs, with $N$ being fixed over all groups. If $N$ is well-chosen, this could balance the bias towards small groups, observed in Figure 10, by making large groups more likely to have small $V_P$ score. However, this would tend to ignore sub-groups of songs, making this analysis less objective.

The second improvement would be to combine these groups with a clustering algorithm in order to reduce the noise created by outliers. In that sense, this method would be more similar to the cluster-based approach of Section 4.2, but where all the clusters must have the same feature values.

The limitation of the feature-based approach pushed for further studies on the relation between $d_s$ and the song features. This lead to the cluster-based approach of Section 4.2 and the neighbour-based approach of Section 4.3. By first using $d_s$ to create relevant groups of songs, and then computing their $V_F$ score, it is possible to highlight inherent properties that were not observable when simply grouping songs according to their feature values (see Figures 13 and 14). When using these two grouping methods, both showed similar results and were able to find groups of songs with repeating feature values that were previously missed. These two methods also complete each other. Since the cluster-based approach creates a partition of the songs, each group can be defined as the representation of a specific pattern structure. Hence, by finding groups with similar feature values, these groups can be interpreted as a typical structure used for this feature value and not elsewhere. Conversely, the neighbour-based approach tends to put some songs into more neighbourhoods than others and cannot be interpreted as typical for the structures. However, it allows the comparison of a song with other similar ones and is able to identify songs with unexpected patterns (see Figure 16).

While the novelty of this study was mainly based on comparing similarity matrices of songs, the direct use of standard notations could open the door to further development in music analysis. First, new types of similarity matrices could be defined. For example, if instead of comparing the notes played between bars, the representation was using the position of the notes in the current chord, then the similarity matrices could contain some finer information on the evolution of the song. In this
Figure 15. Neighbourhoods ordered according to their $V^F$ score with regards to the feature **year**. The blue dots, blue bars, and red bars respectively represent the average year, the deviation around this average year, and the average distance between songs of the neighbourhoods. The yellow star represents the year of the centre song. This approach allows the identification of songs with early use of patterns (such as the first two songs, Def Leppard - Love Bites and Heart - If Looks Could Kill), late use of patterns (such as the third song, The Chainsmokers - Don’t Let Me Down), or on-time use of patterns (such as the fourth song, P!nk - Who Knew). A few neighbours of The Chainsmokers - Don’t Let Me Down are represented in Figure 16.

Figure 16. The representation of the song The Chainsmokers - Don’t Let Me Down, as well as a few chosen neighbours. As it appears from this figure, the pattern shown by The Chainsmokers - Don’t Let Me Down is taken from an early period. Moreover, when comparing the genre of these different songs, it also appears as The Chainsmokers - Don’t Let Me Down does not fit the genres of its neighbours. This is an interesting example of song whose patterns is borrowed from another period of time and style of music.

case, a simple pattern being repeated on different scales could be discovered, and this could lead to a better definition of pattern matrices. Second, the use of music notations could also be useful when
Comparing different songs. Either by directly comparing what is being played, or by comparing chord progressions, a new distance on songs $d_s$ could be defined, not directly related to the pattern structure, but rather to the composition structure. Finally, by only considering a subset of the instruments of the song, this method could be used to characterize songs according to specific metrics (for example the drum patterns, or the rhythmic patterns). Using this idea could also lead to representing songs with multiple similarity matrices, according to the different instruments, and then using more complex statistical tools to study their properties.

This study opens the door to further development in music structure analysis, and especially in the field of similarity matrix representation. With the newly created dataset (available in [5]), made of standard music notations of songs, new types of similarity matrices can be defined, for example by borrowing methods from music theory. The novel approach of this study, based on comparing similarity matrices rather than analyzing them individually, also creates new tools for the analysis of song structure. It is now possible to classify a large number of songs by similarity of patterns and to highlight repeated structures used by artists, years, or genres.

Acknowledgements

This project all started from a discussion with J. Barrett on basic drum beats, and his input made me question the structure of popular songs, which lead to this study. I also wish to thank Dr. Addario-Berry for his support on this project, as well as his two colleagues, Dr. Ouzounian and Dr. Hasegawa, for their input in creating the background of this study. I am also very grateful to N. Kauf, T. Voirand, and T. Stokes for proofreading this work and for their advice.

References

Appendix A. Dataset statistics

The properties of the dataset used for this study can be found in Table 2 and some properties of the features can be found in Table 3. More information can be found in [5].

<table>
<thead>
<tr>
<th>#Songs</th>
<th>#Artists</th>
<th>#Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>4166</td>
<td>1431</td>
<td>6</td>
</tr>
</tbody>
</table>

Table 2. A summary of the statistics of the dataset used for this study.

<table>
<thead>
<tr>
<th>Feature</th>
<th>artist</th>
<th>year</th>
<th>genre</th>
<th>types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of values</td>
<td>1431</td>
<td>62 (1958 - 2019)</td>
<td>473</td>
<td>120</td>
</tr>
</tbody>
</table>

Table 3. A summary of the statistics of the features.