Abstract Feature Detection Across Multiple Viewpoints Via Local Self-Alignment

Anonymized

Abstract

The ability to identify abstract structural features is critical to developing human-level concept models and is lacking with respect to applications designed for complex sequential data. We present a novel approach to discovering structure within artifacts resulting from self-similarity which uses a multiple Smith-Waterman self-alignment. Viewpoint-specific alignment scoring functions enable structure finding in primitive features. These can be compounded to create scoring functions that find higher-level structure. We demonstrate our approach by finding harmonic, pitch, rhythmic, and lyrical structure in symbolic music and then compounding these viewpoints to identify the abstract structure of verse-chorus segmentation.

Introduction

Human-level concept learning relies on the ability to model artifacts at increasing levels of abstraction (Lake, Salakhutdinov, and Tenenbaum 2015). In visual imagery, pixels form strokes which form shapes which form objects. In natural language, letters make words which make phrases which make sentences. Whatever the medium, the ability to learn high-level features is critical to an effective model of the domain, for either discrimination or generation.

Often features of interest are abstract, that is they are not explicitly represented in an artifact description. In poetry or lyrics, features such as rhyme scheme are not usually labeled; however, even beginning readers are capable of identifying intentionally rhymed phrasing (Englemann and Bruner 1974). In music, features such as verse-chorus segmentation and repeated motifs are infrequently labeled but are nonetheless readily inferred by even non-musicians from what *is* represented (e.g., chords, melody). This structure significantly relates to meaning (Nunes, Ordanini, and Valsesia 2015), and although audiences will find structure even where it was not intended, they readily express criticism of artifacts in which they perceive little or no structure. Human-level reasoning about artifacts and domains hinges on the ability to recognize structure within *primitive* features (i.e., features that are labeled) from which *abstract* structural features can then be inferred. Such features are helpful for evaluating, classifying, comparing, and/or generating structured artifacts (Bodily, Bay, and Ventura 2017). In addition, style-transfer and cross-domain translation of ideas is better facilitated by the ability to elucidate abstract structural representation (LeCun, Bengio, and Hinton 2015).

We present a novel approach to inferring abstract structural features. The approach is readily applicable across domains where structure can be modeled in terms of self-similarity (e.g., bioinformatics, natural language, and audio signal processing). As a concrete example for the purposes of demonstration, we examine the inference of abstract structure in lyrical, sectional-form music lead sheets, with the goal of identifying patterns of repetition within viewpoints (c.f., (Conklin and Witten 1995)) and at the more abstract levels of detecting chorus and verse structures.

Identifying *identical* structural patterns is trivial for humans and computers alike. However, humans identify structure extremely well even when repetitions are *not* identical. In music, for example, a human listener readily identifies melodic and harmonic similarity between verses despite variation in pitch, rhythm, and lyrics. Alignment algorithms—such as the Needleman-Wunsch (NW) (Needleman and Wunsch 1970) or Smith-Waterman (SW) (Smith and Waterman 1981) algorithms—have traditionally been used to align similar non-identical sequences in bioinformatics and natural language, although their implementation usually focuses on finding similarity *between* rather than *within* sequences. Furthermore, sequence alignment algorithms have typically been used on sequences of discrete tokens belonging to finite-length alphabets, making it easy to derive static scoring matrices (e.g., PAM (Dayhoff, Schwartz, and Orcutt 1978) and BLOSUM (Henikoff and Henikoff 1992)) for defining a pairwise scoring function.

Many domains, such as graphic art and signal processing, are represented as continuous sequences with abundant features that make enumerating the event space unreasonable. Furthermore, although there is significant interest in finding patterns *between* artifacts in these domains, many important abstract features exist only as a result of patterns *within* single artifacts. Therefore, though sequence alignment may serve well to find regions of self-similarity, several significant adaptations are required to elucidate structure within artifacts in continuous, feature-rich domains.

Methods

The fundamental premise of the approach is that structure in an artifact exists by virtue of self-similarity. In music, the verse-chorus structure is a product of similarity across viewpoints such as melody, chords, and (for choruses) lyrics.

A primary challenge in alignment is determining alignment parameters. Sequence alignment algorithms generally require defining a gap or *insertion/deletion* cost, G, as well as a scoring function $s(x_1, x_2)$ for two arbitrary sequence elements x_1 and x_2. These definitions are non-trivial because they can dramatically affect the resulting alignment.

In traditional alignment domains, the definition of a scoring function is relatively straight-forward because sequence elements are easily represented using a (relatively) small alphabet. In this case the scoring function usually consists of a simple lookup table where values in the table represent the likelihood that one element is aligned with any other element (Henikoff and Henikoff 1992).

However in considering the alignment of musical sequences, a sequence element or *event* is significantly more complex for a few reasons. First, music—both acoustic and symbolic—represents a continuous sequence of sound. It may be discretized at various intervals (e.g., acoustic sampling rates, metrical beats, etc.), but how the sequence is discretized will directly impact the ability to detect patterns across various viewpoints. Because the time and space required per alignment increase exponentially with the sampling rate, we chose a sampling rate of 2 events per beat.

The second complexity involved in a musical sequence element is that, even given a particular discretization of musical events, a single event (e.g., Figure 1) is composed of many different viewpoints. Even if we consider a relatively simple representation of music such as a lyrical lead sheet, combining the number of features to consider per musical event with their respective ranges is sufficient to define a intractable number of unique musical events (Table 1).

In practice our goal is to find structure across a subset of these features. For example, our intuition might suggest that structural motifs in a song's lyrics depend primarily on an event's lyric content, lyric onset information, and measure/beat (for evaluating temporal spacing). We might struggle, however, to know the relative importance of each of these features in scoring similarity. The lyric scoring function must also score pairs of events in which no lyric is sung or perhaps where no note is being played (rest), though intuition about what these scores should be may also be lacking.

Rather than attempting to guess how each of these features should be weighted, we turn to a machine learning solution in which the computer via a genetic algorithm (GA) is able to find the weights for each of these features which, when used in the scoring function, properly aligns musical subsequences that are known to be similar. Each primitive viewpoint requires its own scoring function and its own training of weights. These functions and weights can then be combined to find abstract structural features *across* primitive viewpoints through the definition of a combined alignment scoring function, effectively *learning-to-learn* (Lake, Salakhutdinov, and Tenenbaum 2015).

Multiple Smith-Waterman Self-Alignment

A traditional NW global sequence alignment is a dynamic programming algorithm (Needleman and Wunsch 1970). For a sequence $a = (a_1, \cdots, a_n)$, let $a' = (a_1, \cdots, a_{n-1})$. The optimal score $S(a, b)$ for the alignment of sequences a and b (with lengths $|a|$ and $|b|$) is defined as a function of the optimal scores for subalignments of a and b:

$$S(a,b) = \begin{cases} |a| * G & \text{if } |b| = 0 \\ |b| * G & \text{if } |a| = 0 \\ \max(S(a',b) + G, & \\ \quad S(a,b') + G, & \text{otherwise} \\ \quad S(a',b') + s(a_{|a|}, b_{|b|})) & \end{cases}$$

where G represents the cost of inserting a gap into the alignment and $s(a_{|a|}, b_{|b|})$ represents a *pairwise scoring function* which evaluates to a score representative of the cost of aligning the element $a_{|a|}$ with $b_{|b|}$. Some variations (including our own) differentiate between a *gap open cost*, G_o, and a *gap extend cost*, G_e, where the former is used the first time a gap is inserted and the latter is used for subsequent, consecutive gaps. In this manner the presence and length of a gap can be penalized independently. In practice, a NW alignment sequentially fills in a $(|a| + 1) \times (|b| + 1)$ matrix, M, where the value $M(i, j)$ at the ith row and jth column represents $S((a_1, \cdots, a_{i-1}), (b_1, \cdots, b_{j-1}))$ (where if $i = 0$ or $j = 0$ the corresponding sequence evaluates to the empty sequence). The global alignment score is the value of $M(|a| + 1, |b| + 1)$. The alignment is produced by starting at position $(|a| + 1, |b| + 1)$ and tracing back through the matrix according to the cells which were used (in the max function) in computing the current cell's value: moving diagonally from (i, j) corresponds to aligning a_i with b_j; moving up aligns a_i with a gap; and moving left aligns b_j with a gap. Backtrack continues as long as $i > 0$ and $j > 0$.

The SW local alignment algorithm alters aspects of the NW global alignment algorithm to find the highest scoring subsequence alignment between two sequences (Smith and Waterman 1981). Modifications are primarily threefold. First, $S(a, b)$ is additionally constrained to be non-negative, essentially allowing the algorithm to discover the beginning of the optimal alignment anywhere in the alignment matrix M. Second, the local alignment score (for the optimal local alignment) is the maximum value in M. The row i and column j where this value appears mark the termination of the local alignment. Backtracking proceeds as in the NW algorithm as long as $M(i, j) > 0$.

We are interested in locally aligning musical phrases. We, however, are interested in more than simply the optimal local alignment; we would like to find all significant local alignments. We thus further adapt the SW algorithm to achieve what we call a *multiple Smith-Waterman* (MSW) *self*-alignment. In this variation, we find multiple backtrack points in M. To do this we define a *local maximum threshold*, τ, and a *minimum event match distance*, ϵ, such that $M(i, j)$ is a local maximum iff $M(i, j) \geq \tau$ and $M(i, j) \geq M(k, l)$ for $\forall k = i \pm \epsilon$ and $\forall l = j \pm \epsilon$. Backtracking then proceeds as in the SW algorithm. Because we are doing self-alignment, we need only compute the upper

Event Feature	Description	Range	Feature Value for E in Figure 1
$is\_rest(E)$	$True$ if E occurs during a rest	$[True, False]$	$False$
$pitch(E)$	the MIDI note value being voiced at E	[0,127] ($\varnothing$ if $is\_rest(E)$)	69
$measure(E)$	the measure in which E occurs (0-based)	$\mathbb{Z}_{>0}$	3
$beat(E)$	the offset in beats within measure $measure(E)$ (0-based)	$\mathbb{R}_{>0}$	0.5
$duration(E)$	the duration in beats of the note or rest being voiced at E	$\mathbb{R}_{>0}$	2.5
$is\_note\_onset(E)$	$True$ if the measure and beat of the onset of the note or rest being voiced at E equals $measure(E)$ and $beat(E)$	$[True, False]$	$True$
$lyric(E)$	the lyric being sung at E	Set of all valid syllables $\cup \varnothing$	"try"
$is\_lyric\_onset(E)$	$True$ if the measure and beat of the onset of the lyric being voiced at E equals $measure(E)$ and $beat(E)$	$[True, False]$ ($\varnothing$ if $lyric(E) = \varnothing$)	$False$
$harmony(E)$	the harmony (represented using chord symbols) being voiced at E	Set of all valid chord symbols $\cup \varnothing$	F
$is\_harmony\_onset(E)$	$True$ if the measure and beat of onset of the harmony being voiced at E equals $measure(E)$ and $beat(E)$	$[True, False]$ ($\varnothing$ if $harmony(E) = \varnothing$)	$False$

Table 1: Features for a music sequence event

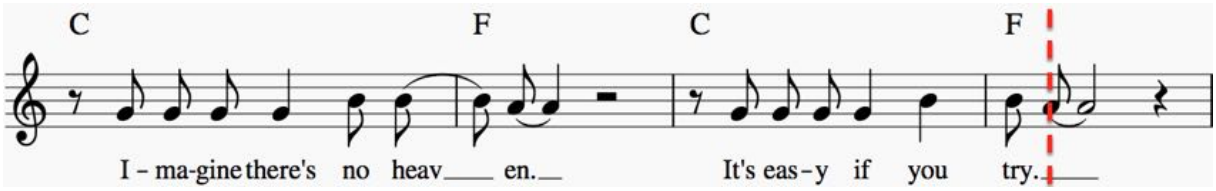


Figure 1: *Example of a music sequence event*. Musical sequences are non-discrete and thus events must be sampled. Table 1 describes the features and feature values for the event sampled at the dotted red line.

diagonal of M (i.e., $j \geq i$). We are also not interested in alignments that are close to the diagonal (i.e., that represent the alignment of an event with itself or close neighbors). We therefore only compute M where $j \geq i + \epsilon$ (see Figure 2). For our implementation, $\epsilon = 4$.

Genetic Algorithm Parameters

Given this general approach, the challenge becomes properly defining the pairwise scoring function $s(a_i, b_j)$ and the general alignment parameters G_o, G_e, and τ. We describe several viewpoint-specific definitions for $s(a_i, b_j)$ below, each of which defines several scoring function parameters. These viewpoint-specific parameters together with the general alignment parameters are learned via GA (see Figure 3).

Initially we generate a population of 20 unique parameterizations where each parameter is randomly initialized in the range [-3,3] (τ is randomly initialized in the range [0,20)). For each of 5000 generations of the GA, we generate 10 new parameterizations via 1) *crossover* of two parameterizations randomly selected from the population and then 2) *mutation* where each parameter has a 0.2 probability of adding a random number in the range [-10,10] to its value (with 0.2 probability τ is multiplied by a factor in the range [0,2)).

Alignment Fitness Function We manually labeled a small subset of the Wikifonia leadsheet dataset with structural repetitions across viewpoints. These labels essentially represent which events we expect to be aligned via our MSW alignment. An event can be aligned with 0, 1, or many other events. We can evaluate a parameterization Γ according to the number of event pair alignments that are true positive (TP), false positive (FP), and true negative (TN) when Γ is used in our scoring function. We do this using the F-score:

$$F_1(\Gamma) = \frac{(1+\beta^2) * TP + 1}{(1+\beta^2) * TP + \beta^2 * FN + FP + 1}$$

with $\beta = 1.0$ to equally weight recall and precision. We add 1 to the numerator and the denominator to ensure that F_1 is defined when no TP are possible (e.g., *Twinkle, Twinkle, Little Star* has no verse). Averaged over all songs in the

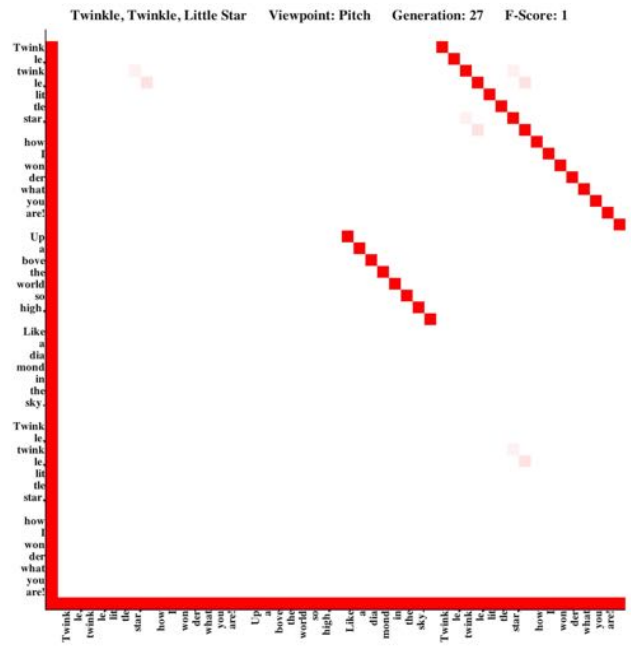


Figure 2: *Finding pitch structure via sequence alignment*. Representing the song *Twinkle, Twinkle, Little Star* as a sequence of discrete events, we align the song against itself using a multiple Smith-Waterman alignment and a pitch-specific pairwise scoring function. The longer red diagonal represents the repetition of pitch between the two choruses in the song whereas the smaller diagonal represents repetition of pitch within the bridge section. Weights for the pairwise scoring function are learned via genetic algorithm (see Figure 3). In this example, 27 generations were required to find weights which maximize the fitness function (F-score).

training data, the F-score represents the fitness of an individual parameterization in our GA. Using this fitness function, we find the optimal parameterization Γ_v^* for each viewpoint v (see Figure 4) via its respective alignment scoring function as described below.

Here, F-score should be viewed as a relative rather than absolute measure of performance for several reasons. First, structure is inherently an abstract concept. This means that what should be labeled in our training data as structure is sometimes ambiguous and can be represented along a spectrum of granularity (e.g., hierarchical rhythmic structure). Second, the scoring functions described below are meant primarily to be illustrative. We found that structure learning is sensitive to which features are included and how they are combined. Third, GAs are stochastic by nature, and the (efficiency of) structure learning is sensitive to this stochasticity. Fourth, we intentionally chose songs with non-trivial structure to see how well this approach would generalize. Thus, even suboptimal F-scores are in many cases reflective of alignments that yield significant structural representation.

Alignment Scoring Functions The intractable range of musical events necessitates defining a dynamic scoring function, meaning a function that does not rely on lookup tables, to determine the similarity of two events. This function should assign a high score to pairs of events that are similar (per viewpoint) and a low score to pairs that are dissimilar.

In our case we define six different scoring functions: one scoring function for each of the primitive viewpoints of harmony, pitch, rhythm, and lyrics, and one scoring function for each of the derived viewpoints representing chorus and verse structures. Each scoring function scores the similarity of two musical events using a unique subset of event features that are indicative of self-similarity in that viewpoint.

Since structural repetitions in music tend to preserve meter, all viewpoint alignment functions are designed to consider the offset within the measure of the two events being aligned. For events e_1 an e_2 we define a beat matching subfunction $M_B(e_1, e_2)$ for this offset alignment as

$$M_B(e_1, e_2) = \begin{cases} \iota_B & \text{if } b_1 = b_2 \\ \Delta_B + \delta_B * |b_1 - b_2| & \text{if } b_1 \neq b_2 \end{cases}$$

where $b_i = beat(e_i)$ and ι_B, Δ_B, and δ_B are weights determined for each viewpoint by the GA.

Harmony A harmony $harmony(e_i)$ represents a set of pitches which we denote $notes(harmony(e_i)) = \{p_1, \cdots, p_n\}$ where each pitch p_i is a MIDI note value modulo 12 to normalize values to a common octave. Using the shorthand $N_i = notes(harmony(e_i))$ we define a harmonic scoring function $S_H(e_1, e_2)$ as follows:

$$S_H(e_1, e_2) = I_H(e_1, e_2) + O_H(e_1, e_2) + M_B(e_1, e_2)$$

with the identity subfunction $I_H(e_1, e_2)$ computed as

$$I_H(e_1, e_2) = \begin{cases} \iota_H & \text{if } N_1 = N_2 \\ \Delta_H + \delta_H / Z(N_1, N_2) & \text{if } N_1 \neq N_2 \end{cases}$$

where the set similarity function $Z(N_1, N_2)$ is defined as

$$Z(N_1, N_2) = (2 * |N_1 \cap N_2| / (|N_1| + |N_2|))$$

Letting $o_i = is\_harmony\_onset(e_i)$,

$$O_H(e_1, e_2) = \begin{cases} \Omega_H & \text{if } o_1 \wedge o_2 \\ \omega_H & \text{if } o_1 \vee o_2 \\ \gamma_H & \text{otherwise} \end{cases}$$

In this manner, ι_H, Δ_H, δ_H, Ω_H, ω_H, and γ_H represent weights to be learned by the GA.

Pitch Letting $r_i = is\_rest(e_i)$ and $p_i = pitch(e_i)$, we compute the pitch score $S_P(e_1, e_2)$ of events e_1 and e_2 as

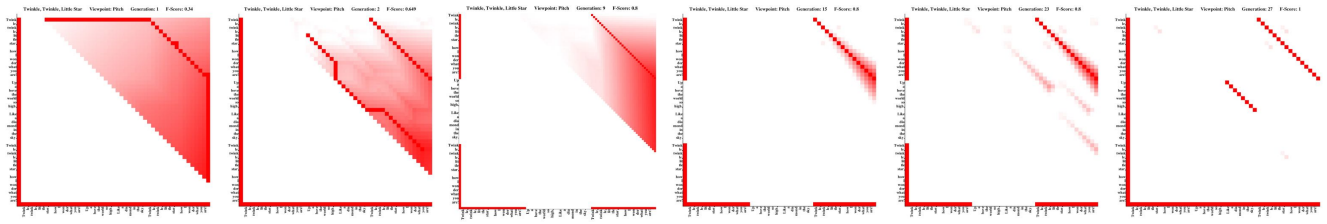
$$S_P(e_1, e_2) = \begin{cases} R & \text{if } r_1 \wedge r_2 \\ \rho & \text{if } r_1 \vee r_2 \\ \gamma_R + M_P(e_1, e_2) & \text{otherwise} \end{cases}$$

with $M_P(e_1, e_2)$ representing the pitch matching subfunction for scoring two events:

$$M_P(e_1, e_2) = I_P(e_1, e_2) + O_P(e_1, e_2) + M_B(e_1, e_2)$$

with

$$I_P(e_1, e_2) = \begin{cases} \iota_P & \text{if } p_1 = p_2 \\ \Delta_P + \delta_P * |p_1 - p_2| & \text{if } p_1 \neq p_2 \end{cases}$$



Generation: 1 $F_1 = 0.34$ | Generation: 2 $F_1 = 0.65$ | Generation: 9 $F_1 = 0.80$ | Generation: 15 $F_1 = 0.80$ | Generation: 23 $F_1 = 0.80$ | Generation: 27 $F_1 = 1.0$

Figure 3: *Learning weights for the pitch scoring function*. As scoring function weights are adjusted via the GA, different alignments result. We use a multiple Smith-Waterman alignment approach to find all local alignments that result in a score above a threshold τ (also determined by the GA). As weightings are found that more accurately align (labeled) pitch repetitions, the F-score increases. Shown is the alignment of the song *Twinkle, Twinkle, Little Star*.

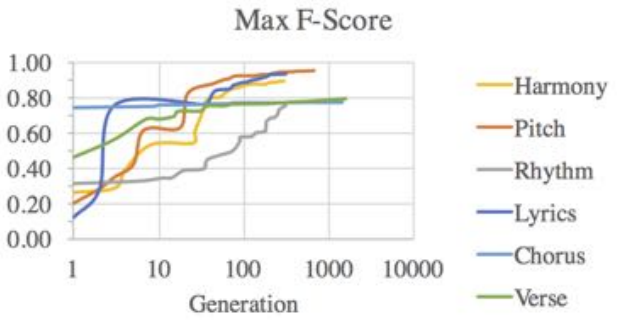


Figure 4: *Learning weights for multiple viewpoints*. Parameterizations for viewpoints were trained via GA for 5000 generations. The graph shows the F-score for the best individual parameterization for each generation (while stil increasing). Rhythm proved more difficult than other primitive viewpoints, likely because rhythm in lyrical music is more readily overlooked when identifying structure than variation in other viewpoints, making it more difficult to distinguish structural patterns from non-structural similarities. The irregular increases in F-score are indicative of the stochasticity of the GA. Because the Chorus and Verse alignment functions use pretrained primitive viewpoint alignment functions, most of the difficult learning for these combinatorial features is done *a priori*, thus requiring fewer generations to find optimal parameterizations.

letting $o_i = is\_pitch\_onset(e_i)$,

$$O_P(e_1, e_2) = \begin{cases} \Omega_P & \text{if } o_1 \wedge o_2 \\ \omega_P & \text{if } o_1 \vee o_2 \\ \gamma_P & \text{otherwise} \end{cases}$$

Again R, ρ, γ_R, ι_P, Δ_P, δ_P, Ω_P, ω_P, and γ_P represent weights to be learned by the GA.

Rhythm Letting $r_i = is\_rest(e_i)$ and $d_i = duration(e_i)$, we compute the melodic rhythm score $S_R(e_1, e_2)$ as

$$S_R(e_1, e_2) = M_R(e_1, e_2) * \\ (I_D(e_1, e_2) + O_P(e_1, e_2) + M_B(e_1, e_2))$$

with $M_R(e_1, e_2)$ representing the rest matching subfunction for scoring two events:

$$M_R(e_1, e_2) = \begin{cases} R & \text{if } r_1 \wedge r_2 \\ \rho & \text{if } r_1 \vee r_2 \\ \gamma_R & \text{otherwise} \end{cases}$$

with

$$I_D(e_1, e_2) = \begin{cases} \iota_D & \text{if } d_1 = d_2 \\ \Delta_D + \delta_D * |d_1 - d_2| & \text{if } d_1 \neq d_2 \end{cases}$$

R, ρ, γ_R, ι_D, Δ_D, and δ_D represent weights to be learned by the GA.

Lyrics Intuitively structural patterns in lyrics are a product of word sequences that repeat. This happens, for example, in choruses or taglines. It does happen on occasion that different iterations of the chorus contain added words or phrases for which some flexibility is needed. Thus we design the lyric scoring function in order to allow the GA to learn appropriate weights for pairs of notes in which one or both notes are either rests or non-lyrical. We also design the lyric scoring function to be able to learn weights which would favor the alignment of lyric onsets.

For two events e_1 and e_2, we compute the lyric score $S_L(e_1, e_2)$. Letting $r_i = is\_rest(e_i)$ and $l_i = lyric(e_i)$,

$$S_L(e_1, e_2) = \begin{cases} R & \text{if } r_1 \wedge r_2 \\ \rho & \text{if } r_1 \vee r_2 \\ N & \text{if } l_1 = \varnothing \wedge l_2 = \varnothing \\ \nu & \text{if } l_1 = \varnothing \vee l_2 = \varnothing \\ M_L(e_1, e_2) & \text{otherwise} \end{cases}$$

with $M_L(e_1, e_2)$ representing the lyric matching subfunction for scoring two events with non-empty lyrics:

$$M_L(e_1, e_2) = I_L(e_1, e_2) + O_L(e_1, e_2) + M_B(e_1, e_2)$$

with

$$I_L(e_1, e_2) = \begin{cases} \iota_L & \text{if } l_1 = l_2 \\ \Delta_L & \text{if } l_1 \neq l_2 \end{cases}$$

Letting $o_i = is\_lyric\_onset(E_i)$,

$$O_L(e_1, e_2) = \begin{cases} \Omega_L & \text{if } o_1 \wedge o_2 \\ \omega_L & \text{if } o_1 \vee o_2 \\ \gamma_L & \text{otherwise} \end{cases}$$

Harmony	*Genetic Representation*: $\Gamma_H = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, \iota_H, \Delta_H, \delta_H, \Omega_H, \omega_H, \gamma_H)$ *Best Solution*: Γ_H^* = (-4, -18, 16.89, 3.23, 3.42, -0.24, 0.39, 1.32, -16.99, -0.97, -10.76, -3.23)
Pitch	*Genetic Representation*: $\Gamma_P = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, R, \rho, \gamma_R, \iota_P, \Delta_P, \delta_P, \Omega_P, \omega_P, \gamma_P)$ *Best Solution*: Γ_P^* = (-18, -11, 27.64, 2, -11.76, -25.79, 3.43, -1.44, -6.88, 4.78, 8.46, -3.53, 3, 0.44, 0.97)
Rhythm	*Genetic Representation*: $\Gamma_R = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, R, \rho, \gamma_R, \iota_D, \Delta_D, \delta_D, \Omega_P, \omega_P, \gamma_P)$ *Best Solution*: Γ_R^* = (-16, -2, 5.84, -5.49, -3.15, -27.99, 0.3, 0.28, 1.38, 10.53, 14.82, -10.9, -4.86, -6.61, -4.9)
Lyric	*Genetic Representation*: $\Gamma_L = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, R, \rho, N, \nu, \iota_L, \Delta_L, \Omega_L, \omega_L, \gamma_L)$ *Best Solution*: Γ_L^* = (-5, -13, 24.1, -2.24, -11.15, -28.18, -4.56, -9.26, 3.11, -10.39, -1.72, -10.99, 7.38, 5.06, 4.83)
Chorus	*Genetic Representation*: $\Gamma_C = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, w_H, w_P, w_R, w_L)$ *Best Solution*: Γ_C^* = (-32, -20, 137.92, 9.05, -11.51, 16.18, 0.43, 3, 0, 4.45)
Verse	*Genetic Representation*: $\Gamma_V = (G_o, G_e, \tau, \iota_B, \Delta_B, \delta_B, w_H, w_P, w_R, w_L)$ *Best Solution*: Γ_V^* = (-7, -17, 313.41, -3.18, 4.18, -3.79, 3, 3.49, 0.07, 1.13)

Figure 5: *Best viewpoint parameterization Γ_v^* for each viewpoint v from the GA after 5000 generations.*

R, ρ, N, ν, ι_L, Δ_L, Ω_L, ω_L, and γ_L are again determined by the GA.

Chorus and Verse Having defined scoring functions for primitive viewpoint alignments, we can now define compound scoring functions for more abstract feature alignment. Consider, for example, that a chorus is generally defined as a musical subsequence in which harmony, pitch, rhythm, and lyrics repeat. A verse is generally defined as a musical subsequence in which harmony, pitch, and rhythm repeat, and lyrics do not repeat. Given that both of these abstract features consider the same set of primitive features, we define a single compound scoring function that can be used (with different parameterizations) to learn structure for both.

For two events e_1 and e_2, we compute a compound alignment score $S_C(e_1, e_2)$ as

$$S_C(e_1, e_2) = w_H * S_H(e_1, e_2) + w_P * S_P(e_1, e_2) + w_R * S_R(e_1, e_2) + w_L * S_L(e_1, e_2)$$

with w_H, w_P, w_R, and w_L representing the weights (to be determined by the GA) of the viewpoints harmony, pitch, rhythm, and lyric respectively, and each of the viewpoint-specific scoring functions as defined above. In learning these abstract features we use optimal parameterizations Γ_H^*, Γ_P^*, Γ_R^*, and Γ_L^* for the subscoring functions as learned on the respective viewpoint-specific alignment tasks (see Figure 5). For learning verse structure, the values of ι_L and Δ_L in Γ_L^* are swapped because Γ_L^* is trained to find where lyrics are similar and verses require lyrics which are different (in similar positions). General alignment parameters G_o, G_e, and τ for subscoring functions are ignored.

Results and Discussion

For each primitive viewpoint v we trained for 5000 generations to find the parameterization Γ_v^* which maximized $F_1(\Gamma_v)$ on the training data (see Figure 5). These parameterizations then are used to identify structure in several songs (see Figure 6). We also tested for how well results generalize to an independent dataset (Table 2).

Each row in Figure 6 effectively represents a 6-faceted structure of a song. Note that within each column, patterns across primitive viewpoints emerge, ultimately combining to yield structural information about abstract features. Notice, for example, how the overlapping across the first 4 columns effectively identifies the choruses of a song whereas overlapping the first 3 and subtracting the 4th effectively identifies the verses. These patterns reinforce the notion that each song has a characteristic abstract structure that is learnable via MSW self-alignment.

	H	P	R	L	C	V
Train	0.90	0.95	0.73	0.82	.79	.75
Test	0.83	0.88	0.66	0.75	.52	.50
Train (hard)	0.90	0.94	0.69	0.89	0.74	.67
Test (easy)	0.84	0.99	0.91	1.00	0.75	1.00

Table 2: *Generalizability*. (Top) Shown are the average F-scores for the training and test sets resulting from a 5-fold cross-validation on our 5-song dataset after 1000 GA generations. (Bottom) Here we show results aggregated from only 2 of the 5 cross-folds in which the holdout song is one of the two simpler compositions (*Twinkle, Twinkle* and *Over the Rainbow*). These results suggest that even with very limited training, generalization is possible, particularly when generalizing to compositions whose complexity is less than or equal to that of the compositions in the training set.

Significant patterns also emerge within columns. Harmonic and pitch structure, for example, tend to show up in longer isolated bands with limited horizontal (or vertical) overlap. Rhythmic structure often shows up as "pyramids" of lines with significant horizontal overlap. These patterns point to the fact that rhythmic structure is far more frequent and even *hierarchical* as compared to structure in other viewpoints. Lyric structure is similar to harmony and pitch structure, but with fewer, sometimes shorter bands. This points to the fact that patterns in harmony and pitch usually span longer ranges within a song whereas lyric patterns are made up of short, dispersed repetitions.

Song-specific and viewpoint-specific structural trends are significant for different reasons. Song-specific trends make it possible to effectively compare the similarity of two songs at an abstract, musical level. This has implications for being able to classify music, recognize different arrangements

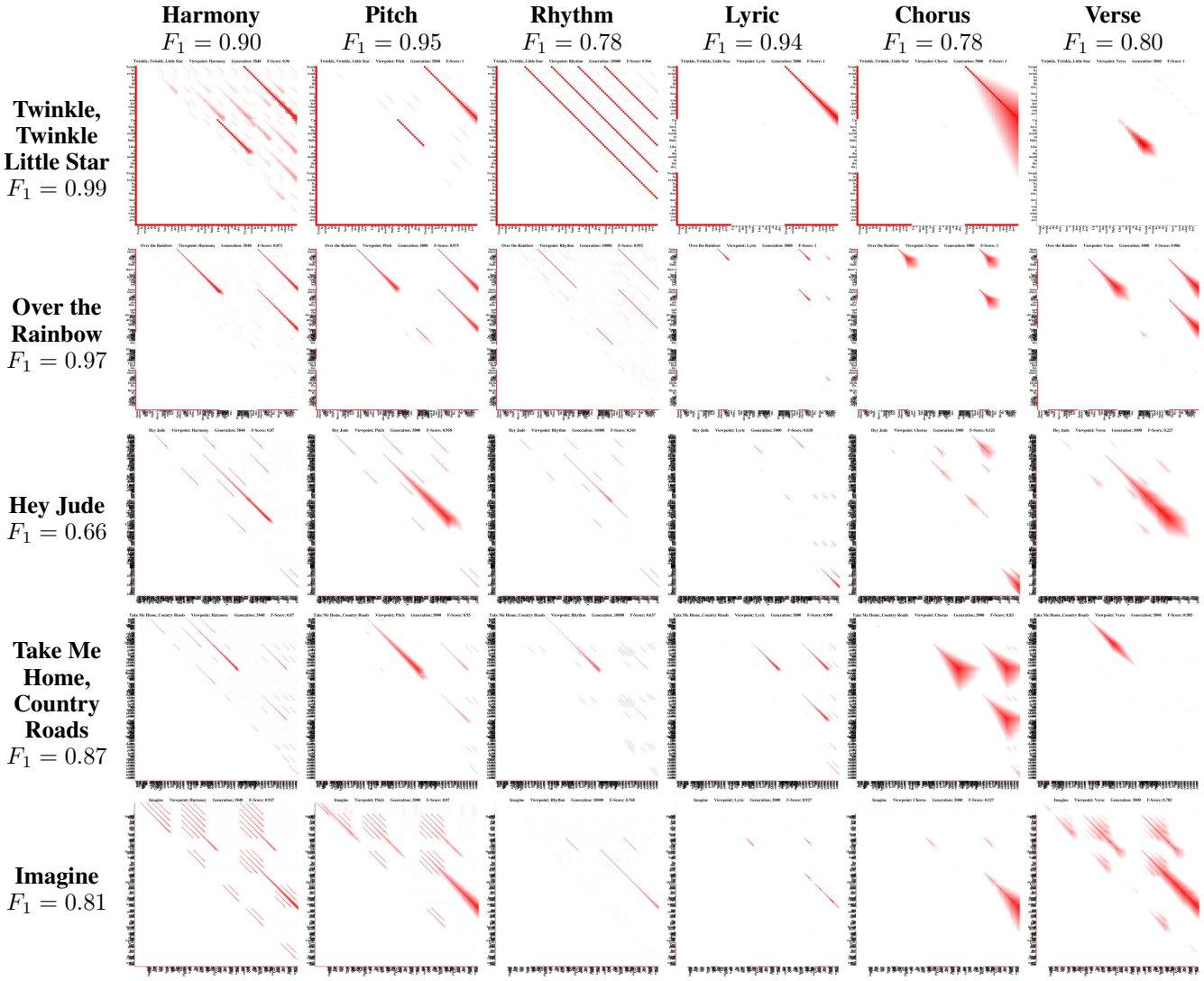


Figure 6: *Structure Detection*. For each viewpoint (i.e., column), the same scoring function weights were used. This suggests a common scoring function can be used to find viewpoint-specific structure across different songs. The *Chorus* and *Verse* columns use scoring functions that are a composite of the four primitive viewpoint scoring functions. Using the GA approach for finding alignment weights for each viewpoint, we can extract the structure for each viewpoint for a given song. These structural representations can then be used for subsequent analyses including classification and generation. For each viewpoint, v, F_1 is $F_1(\Gamma_v^*)$. For each song, F_1 is the average $F_1(\Gamma_v^*)$ across alignments for all viewpoints v for that song only.

of the same song, and recommend music with similar structural elements. Viewpoint-specific trends are significant in being able to generate novel structures for novel music, aiding song-writers and musical metacreationists to discover novel, meaningful structures. These trends have implications for probabilistic parsing, referring to the ability to compute a probability representing how well a musical sequence fits within a particular genre or appeals to a particular audience.

The approach, results, and implications we have demonstrated are not constrained to the symbolic music domain—similar functions, alignments, and patterns can be derived in other domains. For example, MSW self-alignment applied to musical audio signals can be used for chorus-detection, an area that has garnered significant interest (e.g., (Gao and Li 2015)). MSW self-alignment applied to linguistic features of poetry or lyrics can be used for rhyme scheme detection.

The ability to infer abstract structural features, which we have demonstrated using multiple Smith-Waterman self-alignment, is a fundamental component of human-level concept learning. Classification, parsing, and generation using higher-level structures equips computational systems with the ability to analyze artifacts in a way that more closely approaches the underlying meanings and intentions.

References

Bodily, P.; Bay, B.; and Ventura, D. 2017. Computational creativity via human-level concept learning. In *Proceedings of the Eighth International Conference on Computational Creativity*, 57–64.

Conklin, D., and Witten, I. H. 1995. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24(1):51–73.

Dayhoff, M.; Schwartz, R.; and Orcutt, B. 1978. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, volume 5. National Biomedical Research Foundation Silver Spring, MD. 345–352.

Englemann, S., and Bruner, E. 1974. *DISTAR: Reading Level I*. Chicago: Science Research Associates.

Gao, S., and Li, H. 2015. Octave-dependent probabilistic latent semantic analysis to chorus detection of popular song. In *Proceedings of the 23rd ACM International Conference on Multimedia*, 979–982. ACM.

Henikoff, S., and Henikoff, J. G. 1992. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences* 89(22):10915–10919.

Lake, B. M.; Salakhutdinov, R.; and Tenenbaum, J. B. 2015. Human-level concept learning through probabilistic program induction. *Science* 350(6266):1332–1338.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436–444.

Needleman, S. B., and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48(3):443–453.

Nunes, J. C.; Ordanini, A.; and Valsesia, F. 2015. The power of repetition: repetitive lyrics in a song increase processing fluency and drive market success. *Journal of Consumer Psychology* 25(2):187–199.

Smith, T. F., and Waterman, M. S. 1981. Identification of common molecular subsequences. *Journal of Molecular Biology* 147(1):195–197.