Inferring Structural Constraints in Musical Sequences via Multiple Self-Alignment

Paul Bodily (bodipaul@isu.edu)
Computer Science Department
Idaho State University
Pocatello, ID 83209 USA

Dan Ventura (ventura@cs.byu.edu)
Computer Science Department
Brigham Young University
Provo, UT 84602 USA

Abstract

A critical aspect of the way humans recognize and understand meaning in sequential data is the ability to identify abstract structural repetitions. We present a novel approach to discovering structural repetitions within sequences that uses a multiple Smith-Waterman self-alignment. We illustrate our approach in the context of finding different forms of structural repetition in music composition. Feature-specific alignment scoring functions enable structure finding in primitive features such as rhythm, melody, and lyrics. These can be compounded to create scoring functions that find higher-level structure including verse-chorus structure. We demonstrate our approach by finding harmonic, pitch, rhythmic, and lyrical structure in symbolic music and compounding these viewpoints to identify the abstract structure of verse-chorus segmentation.

Keywords: Structural Inferrence; Musical Sequence; Self-alignment

Introduction

Human-level concept learning relies on the ability to model artifacts at increasing levels of abstraction (Lake, Salakhutdinov, & 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. The ability to learn high-level features is critical to an effective model of the domain, either for 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 & 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, & 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 infer *abstract* structural features from looking only at *primitive* features (i.e., features that are labeled). Abstract features are helpful for evaluating, classifying, comparing, and/or generating structured artifacts (Bodily, Bay, & 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, & Hinton, 2015).

Much work in the area of probabilistic constrained modeling has focused on the design of generative models that impose unary and binary structural constraints in sampling sequence solutions (Papadopoulos, Pachet, Roy, & Sakellariou, 2015; Roy et al., 2016; Perez & Régin, 2017). Although the sequential transition probabilities of these models are trained from data, the constraints they impose are generally assumed to be predefined as part of the problem definition, begging the question of how the constraints themselves—and the binary constraints in particular—might be learned from data. Different forms of poetry, for example, may have different rhyme schemes. Rather than rely on manually encoding each structural pattern, how might a system be taught to autonomously recognize these higher-level patterns of repetition? These types of patterns span lengthy regions of text and music and are therefore not readily captured by Markovian processes. For this reason the problem has been referred to as *the long-term dependency challenge* (Collins & Laney, 2017).

Much related work exists for finding long-term dependencies in sequential data. A common approach known as "templagiarism" attempts to elucidate high-level structure from an existing artefact for reuse in the creation of novel artefacts (Pachet, Papadopoulos, & Roy, 2017). Meredith, et al. discover patterns of multidimensional repetition using maximal translatable patterns (MTPs) (Meredith, Lemström, & Wiggins, 2002). Collins, et al. follow up on this work with a pattern discovery algorithm called SIACT to discover translational patterns in baroque keyboard works (Collins, Thurlow, Laney, Willis, & Garthwaite, 2010), which they later use in extracting patterned repetitions in music (Collins & Laney, 2017). Lattner, et al. use bootstrapping in feed-forward neural networks to perform unsupervised melody segmentation (Lattner, Chacón, & Grachten, 2015). Other work has approached the musical sequence segmentation problem using restricted Boltzmann machines (Lattner, Grachten, Agres, & Chacón, 2015).

In contrast to these methods, we turn to a class of algorithms long used for learning sequence structure in bioinformatics. Sequence alignment algorithms—such as the Needleman-Wunsch (NW) (Needleman & Wunsch, 1970) or Smith-Waterman (SW) (Smith & Waterman, 1981) algorithms—have traditionally been used to align genetic sequences, although their implementation usually focuses on finding similarity *between* rather than *within* sequences. 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, & Orcutt, 1978) and BLOSUM (Henikoff & Henikoff, 1992)) for defining a pairwise scoring function.

We present an approach to inferring abstract structural features in music that uses genetic algorithms (GAs) to determine viewpoint-specific scoring functions for structural sequence alignment. 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 musical aspects and at the more abstract levels of detecting chorus and verse structures.

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 & 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).

Multiple Smith-Waterman Self-Alignment

The strength of the NW and SW algorithms is that they do not require exact matches, but rather tolerate some noise while still recognizing structurally similar subsequences. A traditional NW global sequence alignment is a dynamic programming algorithm (Needleman & 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 to find the highest scoring *sub*sequence alignment between two sequences (Smith & Waterman, 1981). Modifications are primarily three-fold. First, $S(a,b)$ is 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 are interested in more than simply the optimal local alignment; we would like to find all significant local alignments.

Table 1: Features for a music sequence event

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 $harmony(E) = \varnothing$)	$False$

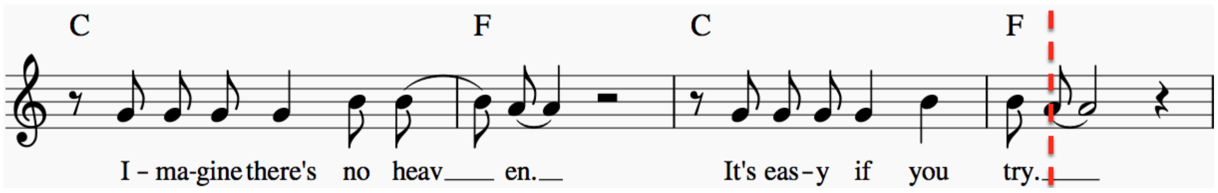

C
F
C
F
I - ma-gine there's no heav___ en.___
It's eas-y if you try.___

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.

We thus 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. 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 $\forall k = i \pm \varepsilon$ and $\forall l = j \pm \varepsilon$. Backtracking proceeds as in the SW algorithm. Because we are doing self-alignment, we need only compute the upper diagonal of M (i.e., $j \geq i$). We are not interested in alignments that are too close to the diagonal (i.e., the alignment of an event with itself). We therefore only compute M where $j \geq i + \varepsilon$ (see Figure 2). For our implementation, $\varepsilon = 4$ (i.e., minimum distance between "motifs" is 2 beats).

Genetic Algorithm Parameters

Here we define 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, along 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 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:

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

with $\beta = 1.0$ to equally weight recall and precision. We add 1 to the numerator and the denominator to ensure F_1 is defined when no TP are possible (e.g., *Twinkle, Twinkle, Little Star* has no verse). Averaged over all songs in the training data, the F-score represents the fitness of an individual pa-

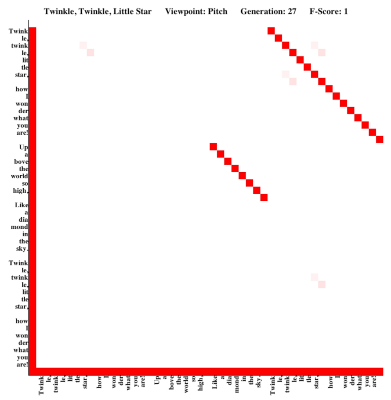


Figure 2: *Finding pitch structure via sequence alignment.* Representing the song *Twinkle, Twinkle, Little Star* as a sequence of discrete events, we align the song against itself using a multiple Smith-Waterman alignment and a pitch-specific pairwise scoring function. The longer diagonal represents the repetition of pitch between two choruses in the song. The shorter diagonal represents repetition of pitch within the bridge section. Weights for the scoring function are learned via GA (see Figure 3). After 27 generations weights were found to maximize the fitness function.

rameterization in our GA. Using this fitness function, we find the optimal parameterization Γ_v^* for each viewpoint v via its respective alignment scoring function as described below.

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 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 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 for $notes(harmony(e_i))$, 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:

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

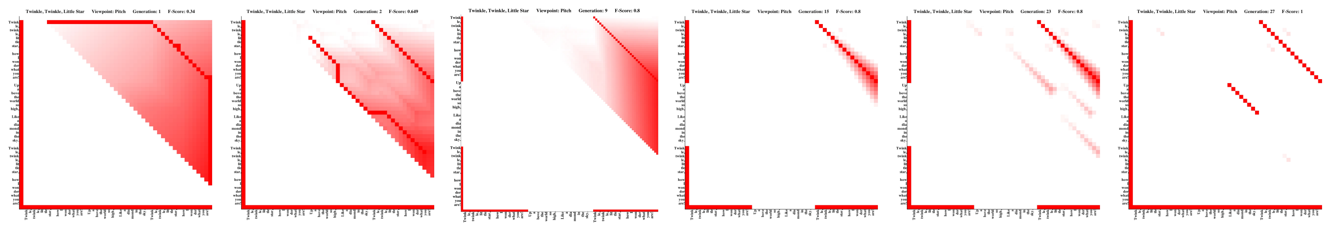
with

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

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.



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

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 are weights 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. Different iterations of the chorus may 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 design the lyric scoring function to learn weights that 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 $o_i = is\_lyric\_onset(E_i)$,

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

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

Chorus and Verse Having defined scoring functions for primitive viewpoint alignments, we can define compound scoring functions for more abstract feature alignment. For example, 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, but lyrics do not. Given 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[1]. For learning verse structure, values of ι_L and Δ_L in Γ_L^* are swapped because Γ_L^* is trained to find similar lyrics and verses contain different lyrics (in similar positions). General alignment parameters G_o, G_e, and τ for subscoring functions are ignored.

[1]learned weights are available upon request

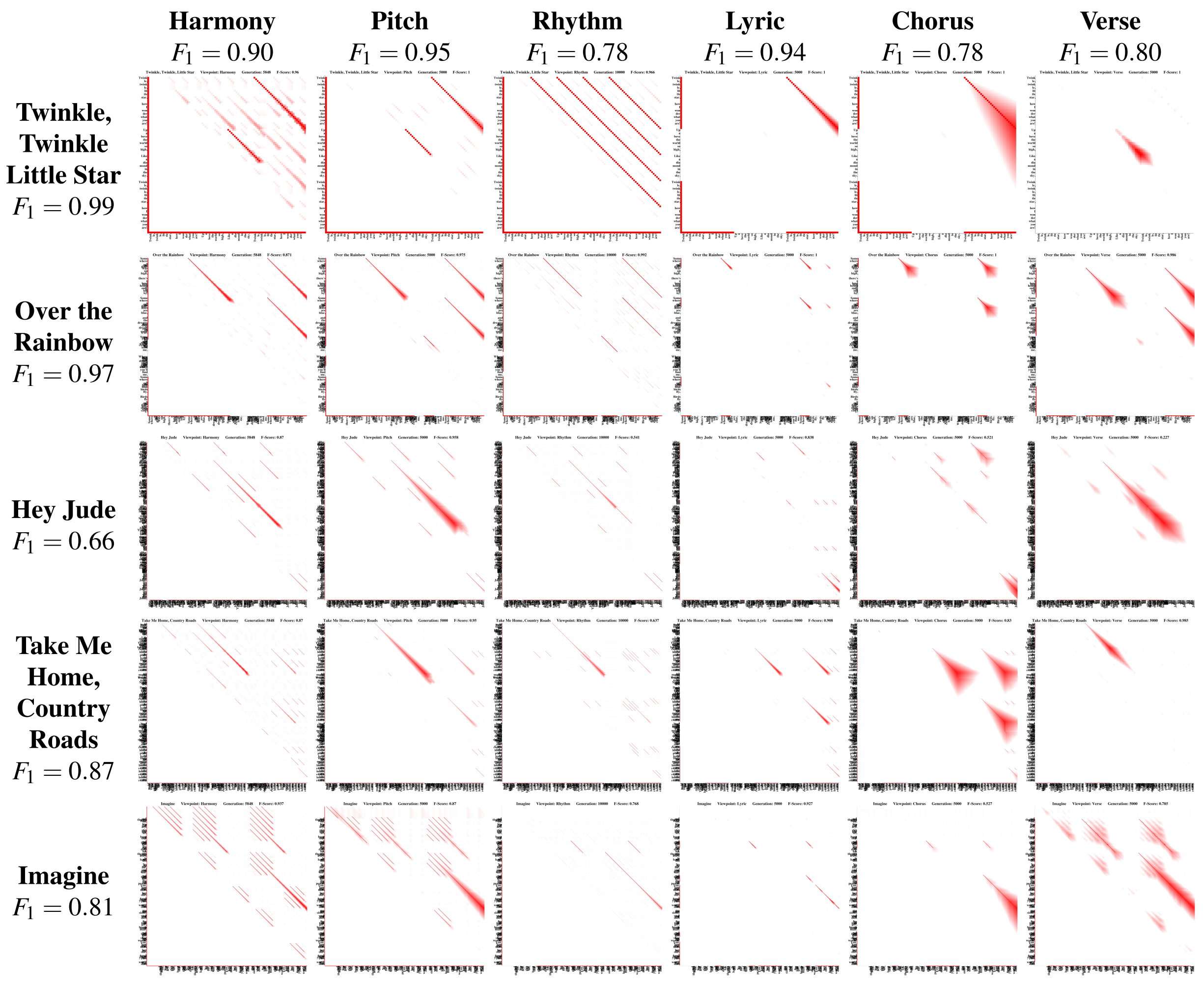


Figure 4: *Structure Detection*. We extract structure for each musical aspect for each song. For each viewpoint (i.e., column), the same scoring function weights were used, suggesting a common scoring function can find structure across songs. The *Chorus* and *Verse* columns use scoring functions composed of primitive viewpoint scoring functions. Average F_1 accuracy scores for each row and column are shown.

Results and Discussion

For each primitive viewpoint v we trained for 5000 generations to find the Γ_v^* which maximized $F_1(\Gamma_v)$ on the training data. These parameterizations are used to identify structure in several songs (see Figure 4). We manually curated and labeled 5 songs composed of varying numbers of (8th-note) musical sequence events: *Twinkle, Twinkle Little Star* (96 events); *Over the Rainbow* (256); *Hey Jude* (562); *Take Me Home, Country Roads* (736); and *Imagine* (448). Together these 5 songs comprise a dataset of 2,098 instances. We tested for how well results generalize to unseen data (Table 2).

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

Significant patterns also emerge within columns. Harmony and pitch, 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 suggest that rhythmic structure is more frequent and *hierarchical* as compared to structure in other viewpoints. Lyric structure is similar to harmony and pitch structure, but with fewer, shorter bands, suggesting 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 of

Table 2: *Generalizability*. (Top) Average F-scores from a 5-fold cross-validation on a song dataset of 2,098 data instances (1000 generations). (Bottom) Results aggregated from 2 of the 5 cross-folds in which the holdout song is of simpler composition (*Twinkle, Twinkle* and *Over the Rainbow*). Even with limited training, generalization is possible, particularly when generalizing to less complex compositions.

	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

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 songwriters 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 & 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 patterns imbues computational systems with the ability to analyze artifacts such as music in a way that more closely approaches their underlying meanings and intentions.

References

Bodily, P., Bay, B., & Ventura, D. (2017). Computational creativity via human-level concept learning. In *Proceedings of the eighth international conference on computational creativity* (pp. 57–64).

Collins, T., & Laney, R. (2017). Computer–generated stylistic compositions with long–term repetitive and phrasal structure. *Journal of Creative Music Systems*, *1*(2).

Collins, T., Thurlow, J., Laney, R., Willis, A., & Garthwaite, P. (2010). A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works. In *Proceedings of the 11th international society for music information retrieval conference* (pp. 3–8).

Dayhoff, M., Schwartz, R., & Orcutt, B. (1978). A model of evolutionary change in proteins. In *Atlas of protein sequence and structure* (Vol. 5, pp. 345–352). National Biomedical Research Foundation Silver Spring, MD.

Englemann, S., & Bruner, E. (1974). *Distar: Reading level i*. Chicago: Science Research Associates.

Gao, S., & 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* (pp. 979–982).

Henikoff, S., & 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., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338.

Lattner, S., Chacón, C. E. C., & Grachten, M. (2015). Pseudo-supervised training improves unsupervised melody segmentation. In *Proceedings of the international joint conference on artificial intelligence* (pp. 2459–2465).

Lattner, S., Grachten, M., Agres, K., & Chacón, C. E. C. (2015). Probabilistic segmentation of musical sequences using restricted boltzmann machines. In *Mathematics and computation in music* (pp. 323–334).

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

Meredith, D., Lemström, K., & Wiggins, G. A. (2002). Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music. *Journal of New Music Research*, *31*(4), 321–345.

Needleman, S. B., & 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., & 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.

Pachet, F., Papadopoulos, A., & Roy, P. (2017). Sampling variations of sequences for structured music generation. In *Ismir* (pp. 167–173).

Papadopoulos, A., Pachet, F., Roy, P., & Sakellariou, J. (2015). Exact sampling for regular and markov constraints with belief propagation. In *Proceedings of the international conference on principles and practice of constraint programming* (pp. 341–350).

Perez, G., & Régin, J.-C. (2017). Mdds: sampling and probability constraints. In *International conference on principles and practice of constraint programming* (pp. 226–242).

Roy, P., Perez, G., Régin, J.-C., Papadopoulos, A., Pachet, F., & Marchini, M. (2016). Enforcing structure on temporal sequences: the Allen constraint. In *International conference on principles and practice of constraint programming* (pp. 786–801).

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